
Spaß mit Computern
Programmieren ganz einfach
Einstieg mit BASIC
CD 60
SONY
Ein Buch von CHIP
Ravensburger
®

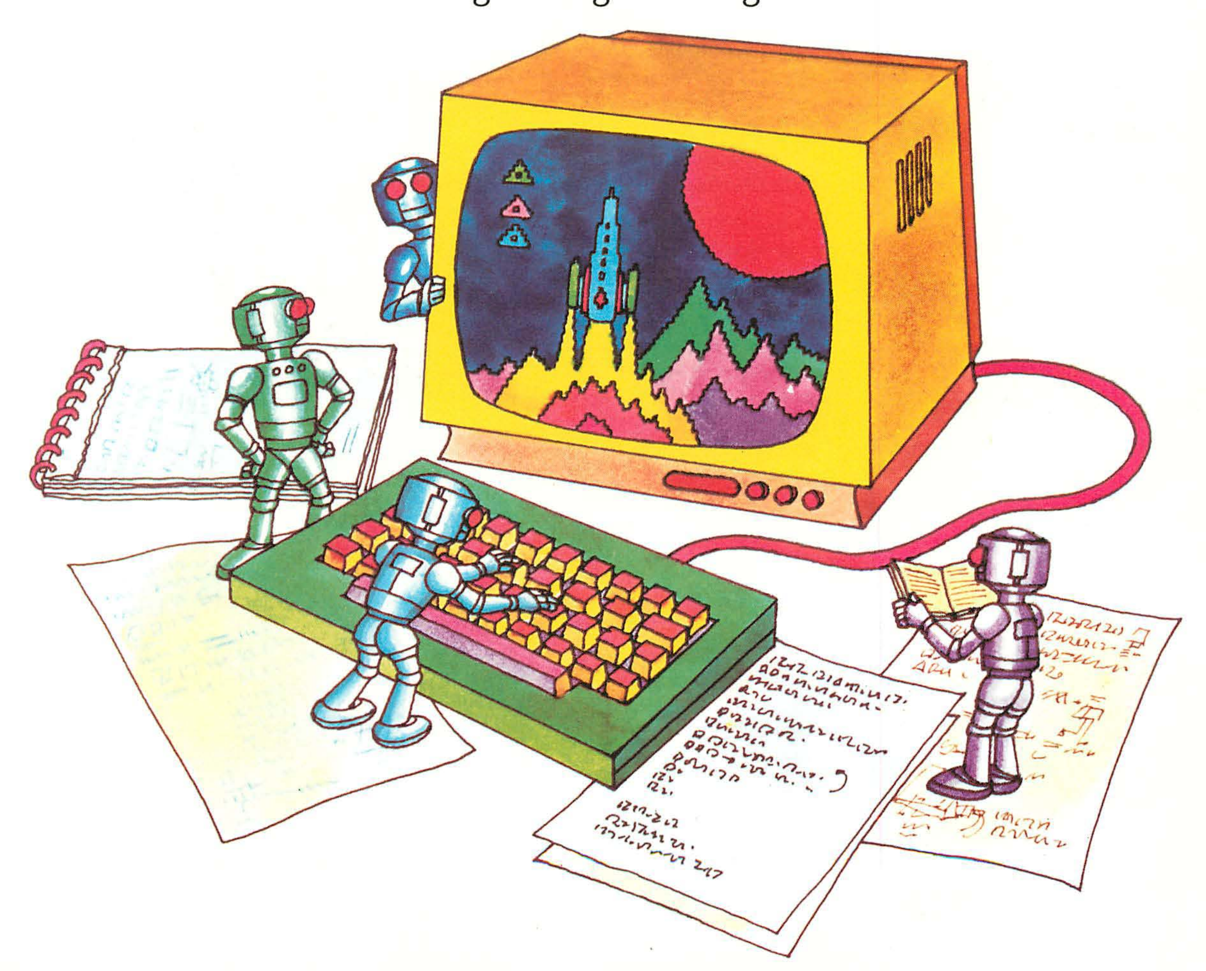
Brian Reffin Smith

# Programmieren ganz einfach

Einstieg mit BASIC

Aus dem Englischen übersetzt von
Barbara Schumacher

Otto Maier Verlag Ravensburg
Vogel-Verlag Würzburg



# Inhalt



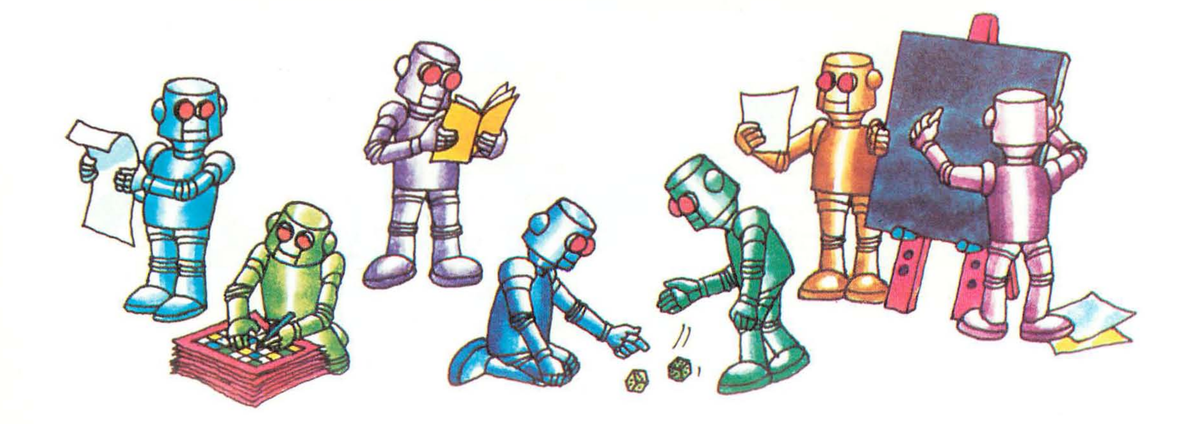

Titel der Originalausgabe: Introduction to Computer Programming
Aus dem Englischen übersetzt von Dr. Barbara Schumacher und bearbeitet von Hubert Hofrat und Martin Stübs
Redaktion der Originalausgabe: Lisa Watts
Illustrationen von Martin Newton und Graham Round
Buchgestaltung: Kim Blundell
Umschlaggestaltung: Achim Köppel
unter Verwendung des Umschlags der Originalausgabe

Gemeinschaftsausgabe:
Otto Maier Verlag, Ravensburg
Vogel-Verlag, Würzburg



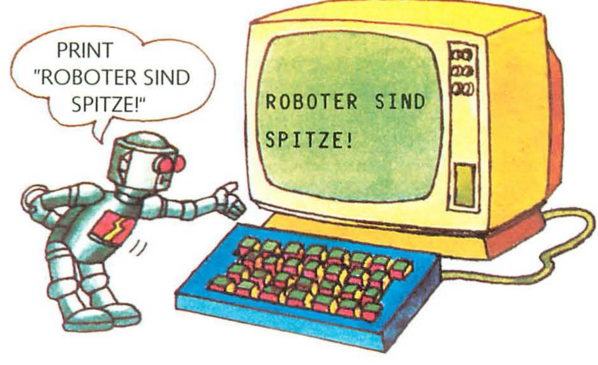


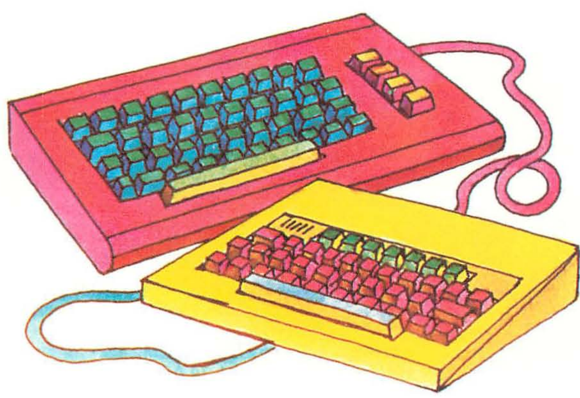
Printed in Belgium
ISBN 3-473-35602-6 (Otto Maier Verlag)
ISBN 3-8023-0765-8 (Vogel-Verlag)

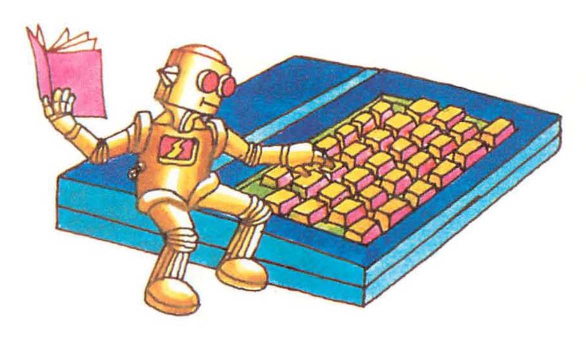
# Über dieses Buch

Dieses Buch ist eine Anleitung für Anfänger, um Computerprogramme in BASIC schreiben zu lernen. BASIC ist die Sprache, die von den meisten Heim-oder Homecomputern verstanden wird. Die Anweisungen werden in einer Form geschrieben, die der Computer verstehen kann.



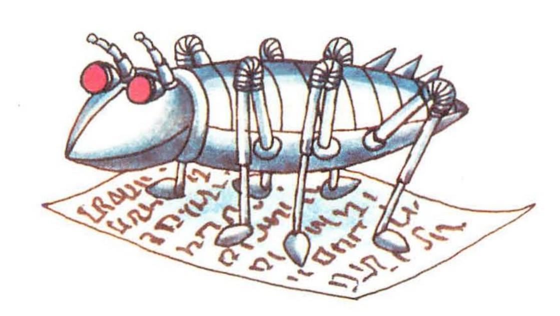



Um mit diesem Buch zu lernen, benötigen Sie keinen Computer. Sie werden die Programme allerdings besser verstehen, wenn Sie diese mit einem Computer ausprobieren können. Die verschiedenen Computertypen verwenden geringfügig unterschiedliche BASIC-Dialekte. Die meisten hier erläuterten Befehle arbeiten auf allen Mikrocomputern; die wenigen Ausnahmen davon sind angegeben.



Zunächst werden einige Hinweise zum Programmieren eines Computers gegeben. Dann werden die wichtigsten BASIC-Befehle schrittweise erklärt. Kurze Programme verdeutlichen, wie sie funktionieren.

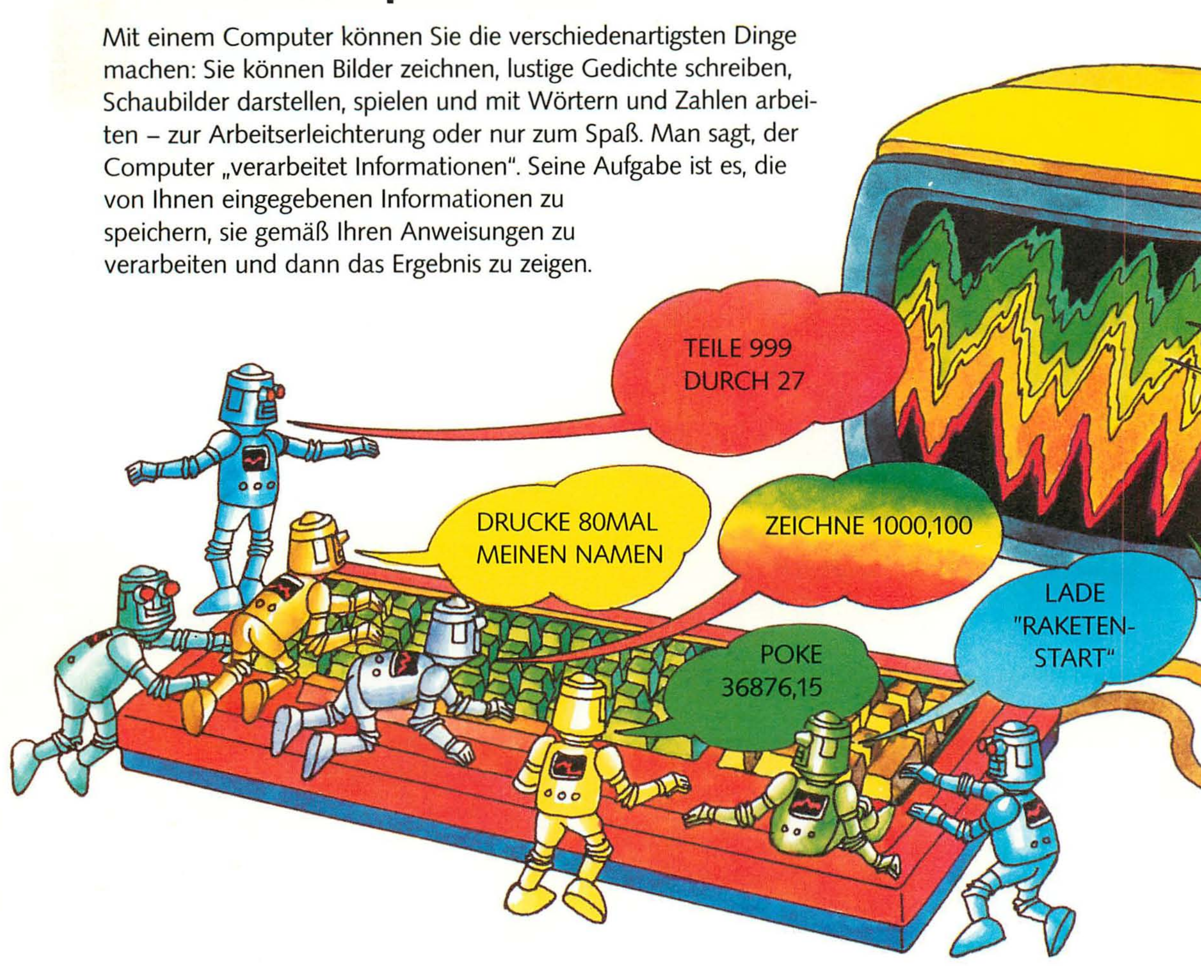
Damit Sie das Programmieren üben können, bietet das Buch Programmierübungen, Rätsel und praktische Änderungsvorschläge für die aufgeführten Programme. Die Lösungen für Programmierübungen und Rätsel stehen auf Seite 44/45.
Am Schluß des Buches sind BASIC-Befehle und Fachbegriffe aufgeführt und kurz erläutert. Dort finden Sie auch Tips und Kniffe fürs Programmieren und schließlich eine Liste von Programmfehlern. Diese „Bugs" (engl. bug = Wanze) stoppen den Ablauf eines Programms.



Wenn Sie einen Computer besitzen, sollten Sie die Programme dieses Buches ausprobieren. Wenn Sie dann mehr über die Arbeitsweise Ihres Computers wissen wollen, schlagen Sie die BASIC-Befehle im Handbuch Ihres Computers nach.
Unter Umständen müssen einige der hier erläuterten Regeln bei Ihrem Computer nicht befolgt werden. Am besten lernen Sie BASIC, wenn Sie eine Vielzahl von Programmen aus Büchern oder Zeitschriften ausprobieren und sie dann geringfügig ab-abändern. Bald werden Sie eigene Programme schreiben können.

# Wie ein Computer arbeitet

Mit einem Computer können Sie die verschiedenartigsten Dinge machen: Sie können Bilder zeichnen, lustige Gedichte schreiben, Schaubilder darstellen, spielen und mit Wörtern und Zahlen arbeiten – zur Arbeitserleichterung oder nur zum Spaß. Man sagt, der Computer „verarbeitet Informationen". Seine Aufgabe ist es, die von Ihnen eingegebenen Informationen zu speichern, sie gemäß Ihren Anweisungen zu verarbeiten und dann das Ergebnis zu zeigen.




Damit der Computer das tut, was Sie wollen, müssen Sie ihm ganz genaue *Anweisungen* oder *Befehle* geben. Eine Folge von Befehlen nennt man *Programm*; die Informationen, die Sie dem Computer geben, sind die *Daten*. Das Programm muß in einer Sprache geschrieben sein, die der Computer verstehen kann, zum Beispiel in BASIC, und es muß den Regeln dieser Sprache entsprechen.

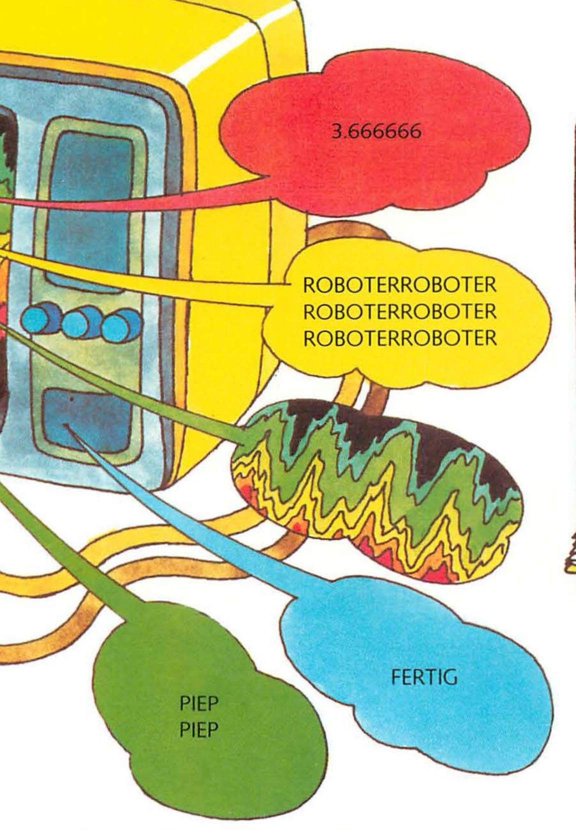
**Mikrocomputer**

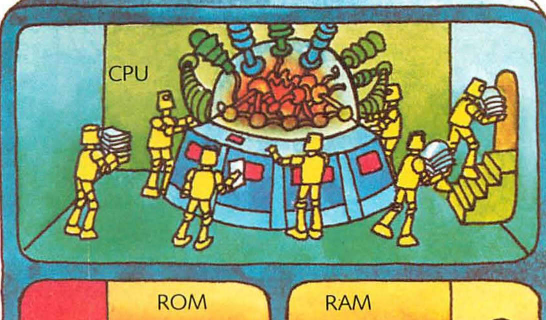
Die meisten Homecomputer bestehen aus einer *Tastatur*, die mit dem Bildschirm eines Fernsehgeräts verbunden ist. Die Anweisungen und Informationen werden über eine Tastatur eingegeben. Alle Eingaben und die vom Computer erstellten Ergebnisse werden auf dem Bildschirm gezeigt.
Manche Computer haben eine Flüssigkristallanzeige, wie ein Taschenrechner. Andere haben einen *Monitor*, einen Bildschirm, der im Gegensatz zu einem Fernsehgerät keine Signale eines Senders empfangen kann.

Das Tastenfeld (oder die Tastatur) entspricht dem einer Schreibmaschine, es hat aber noch zusätzliche Tasten. Bei manchen Computern kann mit einer Taste ein ganzer BASIC-Befehl eingegeben werden; er muß dann nicht Buchstabe für Buchstabe getippt werden.

## Das Innenleben des Computers

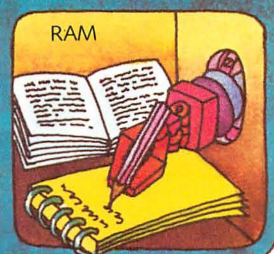
Ein Computer hat zwei wichtige Funktionseinheiten: Der *Mikroprozessor* oder die CPU (engl. **C**entral **P**rocessing **U**nit) ist das Steuer- und Rechenwerk. Im *Speicher* werden Programme und Daten abgelegt.










Genauer gesagt hat der Computer zwei Speicher: Im *Festwertspeicher*, dem ROM (engl. **R**ead **O**nly **M**emory), ist das Programm für die Funktionen des Mikrocomputers abgespeichert. Der *Schreib-/Lesespeicher*, das RAM (engl. **R**andom **A**ccess **M**emory), wird zur vorübergehenden Speicherung von Programmen, Befehlen und Daten benötigt, die über die Tastatur eingegeben werden. Dieser Speicher wird beim Abschalten des Computers gelöscht.

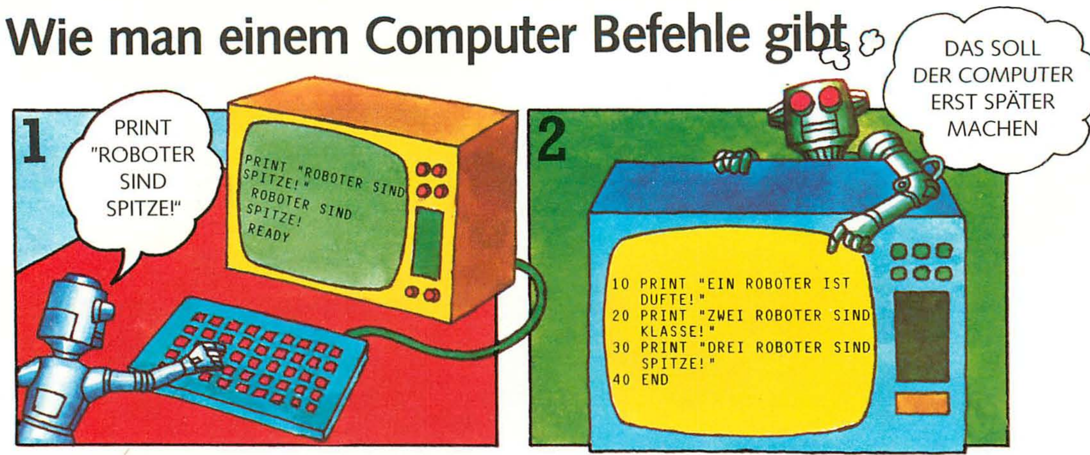


Drucker

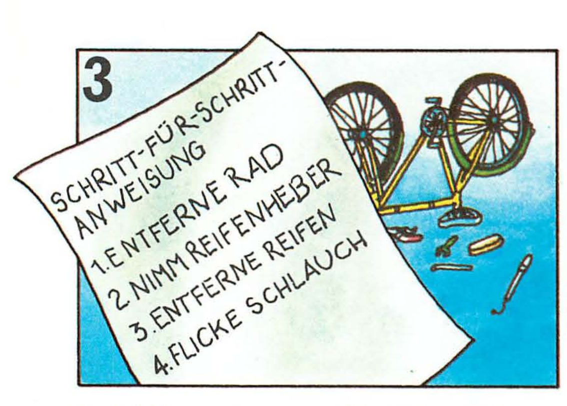
Die Informationen werden in der Regel auf dem Bildschirm ausgegeben. Sie können auch auf einem Drucker ausgegeben werden. Dies ist nützlich, da sonst die Informationen beim Abschalten der Geräte verlorengehen.

Die Informationen können auch mit einem Kassettenrecorder gespeichert werden. Programme und Daten werden auf Magnetband gespeichert und können bei Bedarf wieder in den Computer eingegeben werden.

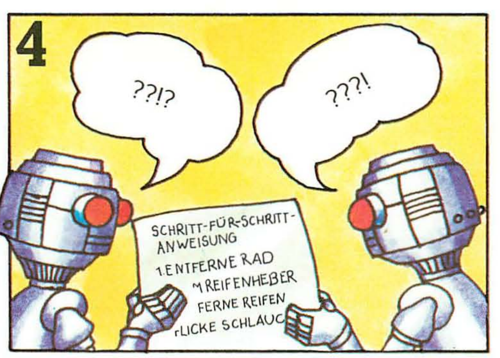
# Wie man einem Computer Befehle gibt



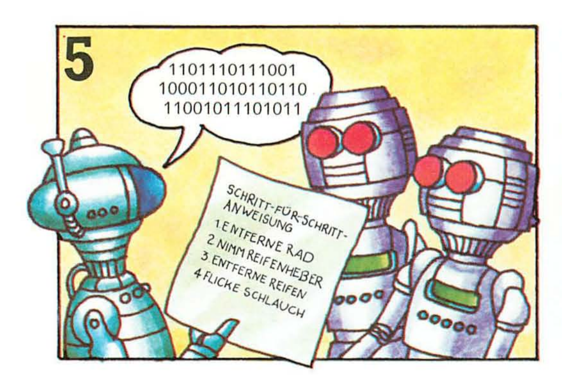

Damit ein Computer etwas ausführt, muß man ihm Befehle in einer Sprache geben, die er versteht. Dies kann ein Befehl sein, der sofort ausgeführt wird, oder ein sogenanntes Programm, eine Folge von Befehlen, die zunächst gespeichert und erst nach dem Start-Befehl ausgeführt werden.



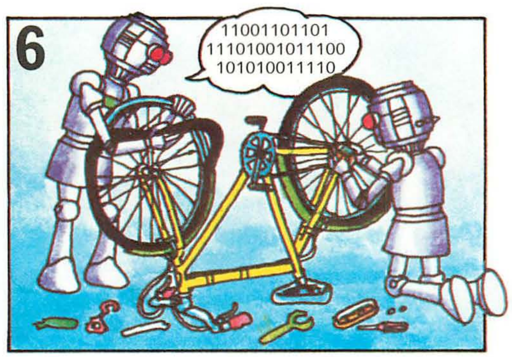

Die Befehle müssen sehr sorgfältig erstellt werden. Der Computer versucht stur, sie auszuführen, auch wenn sie falsch sind.




Er versteht keine Befehle in unserer Sprache, daher müssen sie in einer Computersprache geschrieben werden. Auf der folgenden Seite sind einige Programmiersprachen beschrieben.




Der Computer arbeitet aufgrund von winzigen Stromimpulsen. Die Befehle werden durch ein spezielles internes Programm in die Maschinensprache, den sogenannten Maschinencode, umgesetzt.

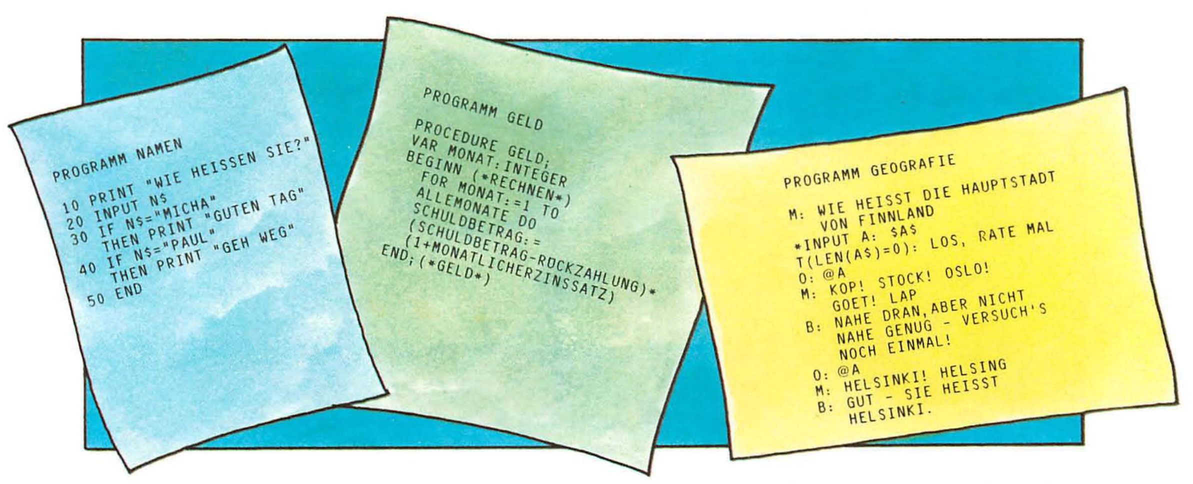



Im Maschinencode wird jede Information durch ein Muster von Impulsen dargestellt. Dieser Code kann mit den Ziffern 1 (Impuls) und 0 (kein Impuls) dargestellt werden.

## Computersprachen

Programme können im Maschinencode geschrieben werden, dies ist aber sehr umständlich. Deswegen wurden spezielle, sogenannte höhere Computersprachen entwickelt, die der Computer in seinen Maschinencode übersetzt.

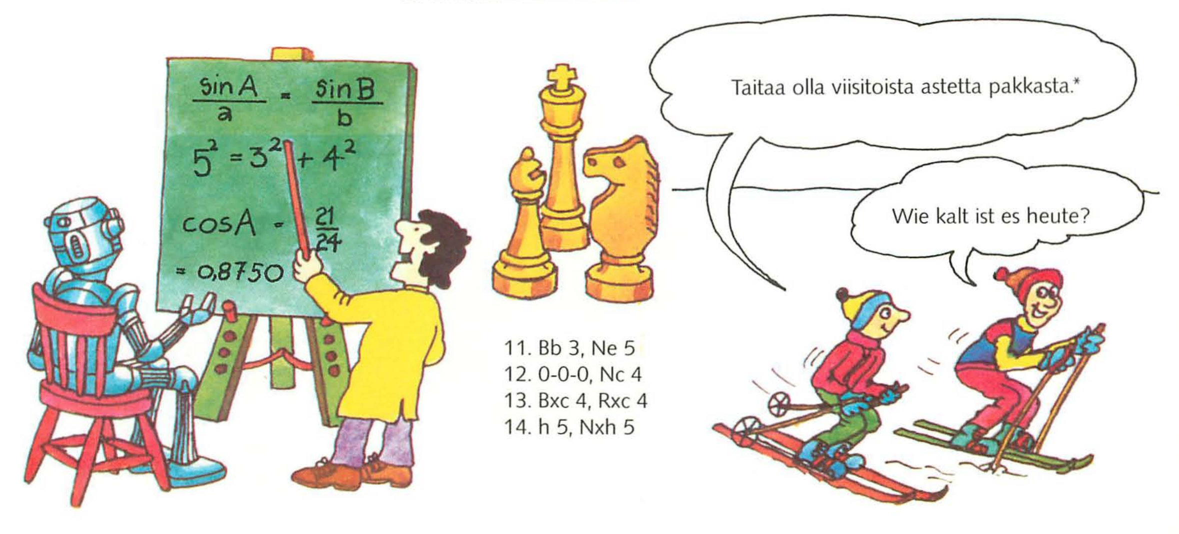
Es gibt eine Vielzahl von höheren Computersprachen, zum Teil wurden sie nur für einen ganz bestimmten Anwendungszweck entwickelt. BASIC ist eine der weitestverbreiteten Sprachen. BASIC bedeutet **B**eginners' **A**ll purpose **S**ymbolic **I**nstruction **C**ode, also Allzweck-Symbolsprache für Anfänger. Doch nicht nur Anfänger verwenden BASIC. Hier drei Beispiele für verschiedene Computersprachen.




Dies ist ein kleines Programm in BASIC. Zeile 20 befiehlt dem Computer, „Wie heißen Sie?" auf den Bildschirm zu schreiben. Der Computer speichert Ihre Antwort und übermittelt Ihnen eine Botschaft, falls Sie z. B. Hans oder Micha heißen.

Diese Sprache heißt Pascal, zu Ehren des französischen Mathematikers Blaise Pascal (1623 – 1662). Der Ausschnitt stammt aus einem Programm, das Finanzprobleme löst. Viele Leute meinen, gute und klare Programme seien in Pascal leichter zu schreiben als in BASIC.

Diese Sprache heißt PILOT. Sie wird für Unterrichts- und Lernprogramme benutzt. Der Computer kann in dieser Sprache Antworten auch dann erkennen, wenn sie nicht ganz korrekt sind.




Zunächst erscheinen Computersprachen seltsam und schwierig, wie das Finnisch in unserem Beispiel rechts. Es gibt eine Vielzahl von Problemen, bei denen spezielle Sprachen angewandt werden.

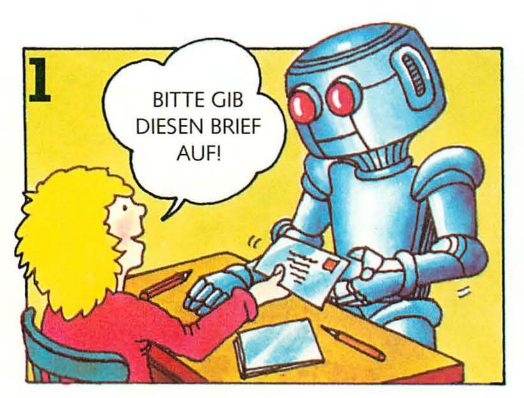
Mathematische Formeln werden zum Beispiel mit speziellen Zeichen und Begriffen geschrieben, auch für Schachzüge oder für Musik werden ganz bestimmte Begriffe und Zeichen verwendet.

* Ich schätze, so minus 15 Grad.

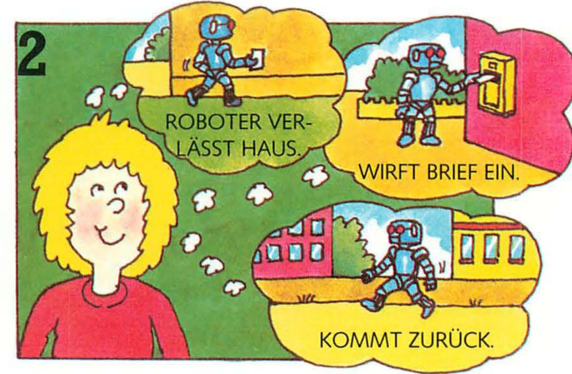
# Wie man ein Programm schreibt

Ein Computerprogramm ist wie eine Spielregel oder ein Kochrezept. Sind diese falsch, kann man das Spiel nicht richtig spielen, oder der Kuchen mißlingt. Ebenso sind die Ergebnisse, die ein Computer liefert, von den Anweisungen abhängig, die Sie ihm geben. Bevor Sie ein Programm schreiben, müssen Sie sorgfältig den Ablauf strukturieren und die einzelnen Schritte untersuchen, die zum Erreichen des Ziels notwendig sind.

**Programm "Brief aufgeben"**




Versuchen Sie, ein Programm zu entwickeln, das einen Roboter veranlaßt, einen Brief aufzugeben. Die einfache Anweisung, wie sie in diesem Bild gegeben wird, kann der Computer nicht verstehen.




Sie müssen genau ausarbeiten, was der Computer zu tun hat, um den Brief aufzugeben. Dabei muß dem Computer jeder Schritt einzeln vorgegeben werden.

**3**

Haus verlassen
- Geh zur Haustür
- Öffne Tür
- Geh hinaus und schließe Tür
- Suche Briefkasten

Brief einwerfen
- Stecke Brief in Schlitz
- Laß Brief los

Heimkommen
- Kehre um
- Geh Weg zurück
- Öffne Tür
- Tritt ein
- Schließe Tür

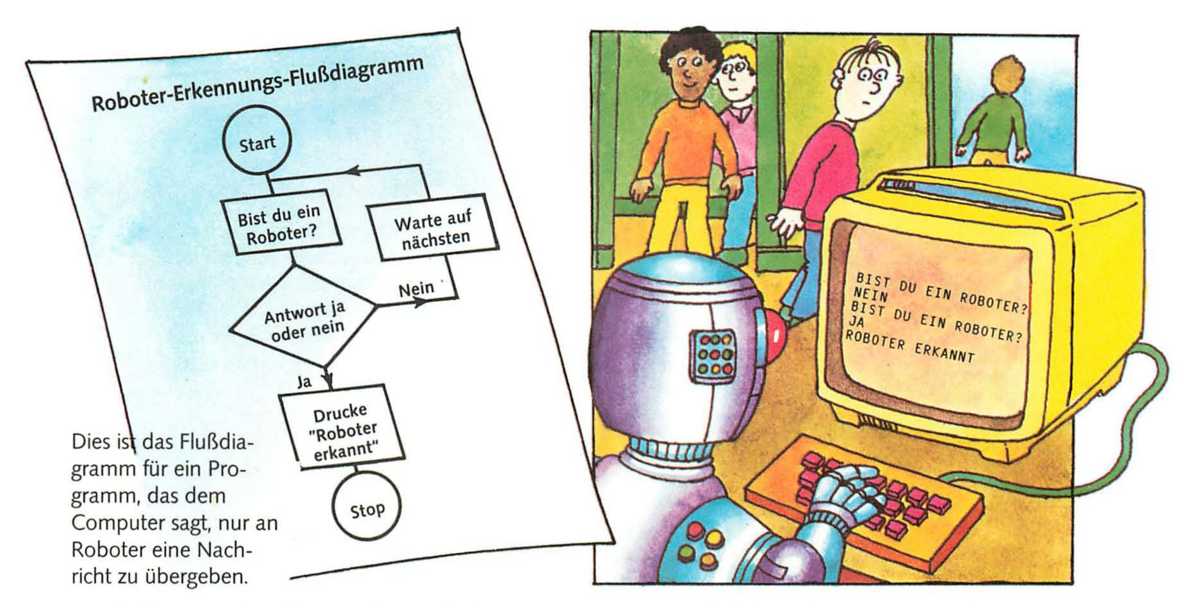
ICH MUSS DEN BRIEF EINWERFEN.

Um das Programm zu schreiben, müssen Sie die Anweisungen in einzelne Schritte zerlegen. Diese Schrittfolge muß in eine Sprache übersetzt werden, die der Computer versteht.

Der Roboter versucht, Ihren Anweisungen zu folgen, auch dann, wenn sie unvollständig oder falsch sind. Die Fehler in einem Programm nennt man „Wanzen" (engl. bug = Wanze); sie führen manchmal zu unerwarteten Ergebnissen – wie das Bild zeigt.

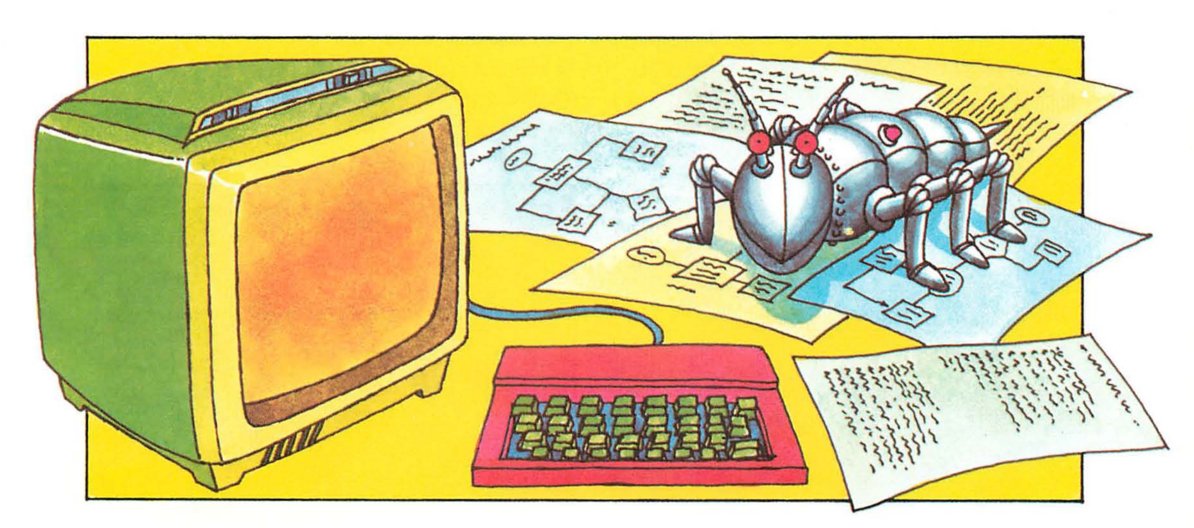
## Flußdiagramme

Bevor man ein Programm schreibt, ist es oft hilfreich, sich ein Diagramm zu zeichnen, das die wesentlichen Schritte zur Lösung einer Aufgabe darstellt. Ein solches Diagramm heißt *Flußdiagramm.* Es zeigt jeden Schritt, den der Computer ausführen muß, und die richtige Reihenfolge dieser Schritte.




Dies ist das Flußdiagramm für ein Programm, das dem Computer sagt, nur an Roboter eine Nachricht zu übergeben.

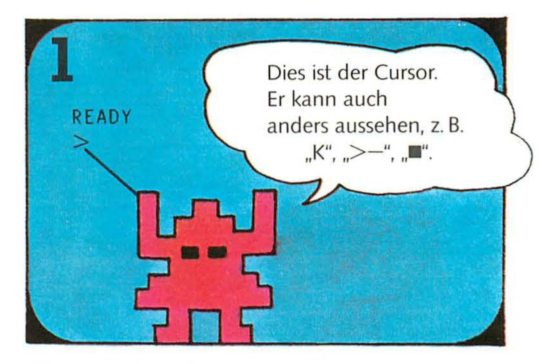
Ein Flußdiagramm besteht aus Feldern, die für die verschiedenen Programmschritte unterschiedlich aussehen. Start und Ende eines Programms haben runde Felder, die Anweisungen an den Computer stehen in rechteckigen Feldern, und die Entscheidungsfelder, bei denen der Computer sich entsprechend der erhaltenen Information entscheiden muß, haben die Form einer Raute. Die Linien zeigen die verschiedenen Wege, denen der Computer folgen kann.



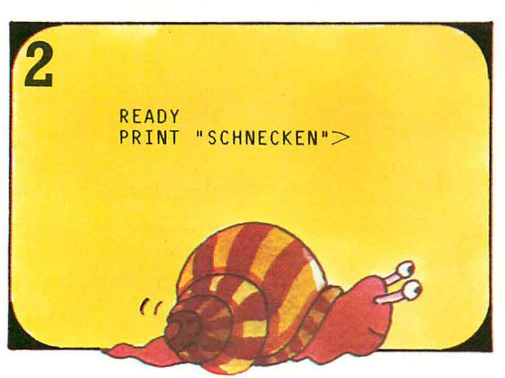
Nachdem alle Einzelheiten des Programms gut überlegt sind, kann man das Programm in BASIC übersetzen und dann auf dem Computer testen. Wahrscheinlich wird dieses Programm nicht sofort laufen, weil es noch einige Fehler enthält. Diese Fehler können Tippfehler sein, die beim Eingeben des Programms in den Computer entstanden sind; es können auch logische Fehler im Programm sein (siehe dazu Seite 42/43). Manchmal verursacht ein Fehler ein etwas anderes Ergebnis, das vielleicht sogar günstig ist. Solche nützlichen Fehler heißen „Möpse" (engl.: pugs).

# Erste Schritte in BASIC

Die meisten BASIC-Ausdrücke sind Wörter in englischer Sprache. Mit einigen Englisch-Kenntnissen kann man leicht erraten, was sie bedeuten. So bedeutet z. B. PRINT: „Zeige auf dem Bildschirm", RUN: „Führe das Programm aus" und INPUT: „Gib dem Computer Information". Auf diesen beiden Seiten wird der Gebrauch von PRINT erklärt. In den meisten Heimcomputern ist ein Interpreter für BASIC bereits eingebaut. Wenn man ein solches Gerät einschaltet, kann man es sofort in BASIC programmieren. Bei manchen Computern muß man allerdings erst ein spezielles Programm mittels Kassettenband eingeben, bevor sie BASIC verstehen.



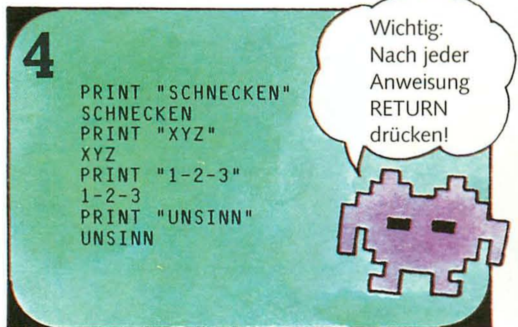

Wenn man den Mikrocomputer einschaltet, werden normalerweise automatisch einige Wörter auf dem Bildschirm erscheinen, zusammen mit einem kleinen Zeichen, das *Cursor* genannt wird. Der Cursor zeigt, an welcher Stelle der nächste Buchstabe erscheinen wird, den man eintippt.




Der Computer läßt Wörter auf dem Bildschirm erscheinen, wenn man PRINT und anschließend in Anführungszeichen den gewünschten Text eingibt. So bedeutet z. B. PRINT "SCHNECKEN", daß der Computer das Wort SCHNECKEN auf dem Bildschirm erscheinen lassen soll.




Der Computer führt Anweisungen erst dann aus, wenn man NEWLINE (RETURN oder ENTER – dies ist von Computer zu Computer unterschiedlich) eingibt.




Der Computer läßt alles das auf dem Bildschirm erscheinen, was man – in Anführungszeichen eingeschlossen – eingibt. Dies können sowohl Buchstaben als auch Zahlen, Wörter oder Zeichen sein. Die Anführungszeichen selbst erscheinen nicht.

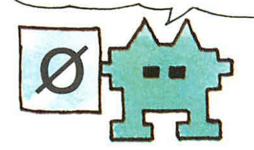



Wenn man nur Zahlen auf dem Bildschirm zeigen will, braucht man keine Anführungszeichen zu benutzen. Der Bildschirm wird wieder frei, wenn man CLS eingibt. (Bitte im Handbuch für Ihren Computer nachprüfen!)

## Ein Programm in BASIC

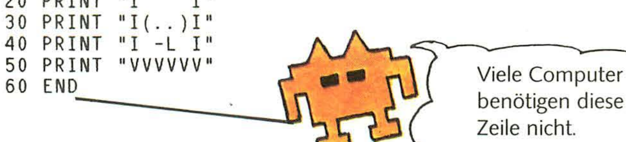
In einem Programm beginnt jede Programmzeile mit einer Nummer. Dies veranlaßt den Computer, die Informationen in seinem Speicher abzulegen und sie nicht eher auszuführen, bis man ihn dazu auffordert. Auf der gegenüberliegenden Seite hatten die Anweisungen an den Computer keine Nummern, daher wurden sie sofort ausgeführt. Hier ist ein kleines Programm, das Zeichen in Form eines Gesichts auf dem Bildschirm erscheinen läßt.




```
10 PRINT "//////"
20 PRINT "I    I"
30 PRINT "I(..)I"
40 PRINT "I -L I"
50 PRINT "VVVVVV"
60 END
```

Die Zeilennummern sind meistens Zehnerzahlen. Man kann dann leicht weitere Programmzeilen hinzufügen, ohne alle Zeilen umnumerieren zu müssen.




Wenn man ein Programm eingibt, muß man RETURN (oder das entsprechende Wort) am Ende jeder Zeile drücken. Die Zeilen erscheinen auf dem Bildschirm, der Computer führt die Anweisungen jedoch erst dann aus, wenn man RUN eingibt.

Achtung: Nicht den Buchstaben O mit der Zahl 0 (Null) verwechseln, dies führt zu Fehlern! Die meisten Mikrocomputer haben eine DELETE-(Lösch-)Taste, mit der man Tippfehler verbessern kann.




```
RUN
MISSING"
LIST
```

Fehleranzeige

Wenn man das Programm nochmals sehen will, LIST (und RETURN) eingeben.

Sobald alle Zeilen eingegeben sind, muß sorgfältig geprüft werden, ob keine Fehler mehr vorhanden sind. Der Computer bearbeitet das Programm, wenn man erst RUN und dann RETURN eingibt.

Wenn das Programm nicht läuft oder das Bild nicht richtig aussieht, muß man das Programm nochmals anschauen und auf Fehler überprüfen. Dazu LIST eingeben. Der Computer gibt dann vielleicht eine Fehlermeldung.

### 1 So berichtigt man Programme

```
RUN
MISSING"
LIST
10 PRINT "//////"
20 PRINT "I    I"
30 PRINT "I(..)I"
40 PRINT "I -L I"
50 PRINT "VVVVVV
60 END
```

Fehlendes Anführungszeichen




Bei den meisten Fehlern gibt der Computer eine Fehlermeldung. Die Fehlermeldungen werden im Computer-Handbuch erklärt. Am einfachsten korrigiert man einen Fehler, indem man die ganze Zeile nochmals eingibt. Der Computer ersetzt dann die alte Zeile durch die neue. Um eine Zeile ganz aus dem Programm zu entfernen, muß man die Zeilennummer und dann RETURN eingeben. Die Möglichkeiten, Zeilen zu korrigieren oder zu verändern, sind von Computer zu Computer unterschiedlich; dazu verwendet man EDIT oder COPY, entsprechend den Erläuterungen im Computer-Handbuch.

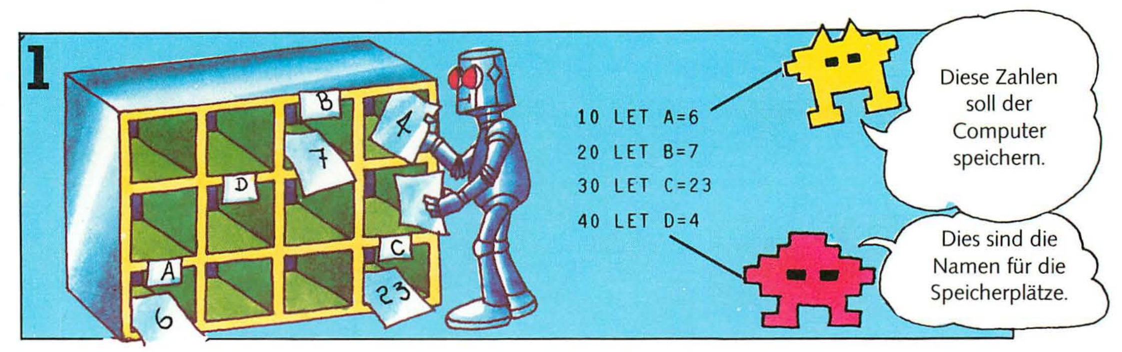




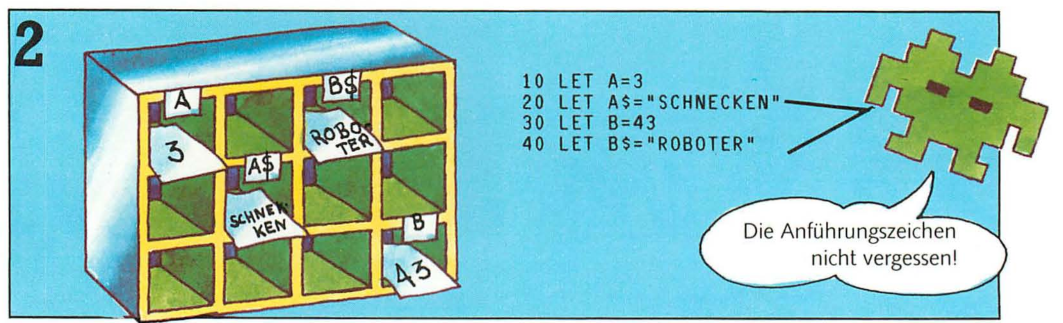
**Programmierübung:** Versuchen Sie, das Programm abzuändern, um dem Gesicht unterschiedliche Ausdrücke zu geben.

# Informationen für den Computer

Der Computer kann natürlich viel nützlichere Dinge tun, als lediglich Figuren auf dem Bildschirm erscheinen zu lassen. Dazu ist es allerdings erforderlich, daß er Informationen oder *Daten* erhält, die er bearbeiten kann. Der Computer speichert diese Informationen in seinem Speicher, bis er Anweisungen erhält, die Informationen zu bearbeiten.




In den Speicher des Computers eingegebene Daten müssen eine Bezeichnung bekommen, damit man sie wiederfindet. Dazu kann man Buchstaben verwenden. Um eine Zahl in einen bestimmten Speicherplatz einzugeben, benutzt man das Wort LET, wie oben gezeigt. Ein gekennzeichneter Speicherplatz heißt *Variable*, da er innerhalb des Programms verschiedene Daten zu verschiedenen Zeiten enthalten kann.




Zum Speichern von Buchstaben und Symbolen in den Speicherplätzen benutzt man eine andere Bezeichnung. Buchstaben und Symbole heißen *Zeichenketten* (engl. strings). Um sie zu bezeichnen, benutzt man die Buchstaben des Alphabets zusammen mit dem Dollar-Zeichen, z. B. C$ (gesprochen „C Dollar").

Zur Speicherung eines Wortes im Speicherplatz wird LET in derselben Weise angewandt wie bei Zahlen-Variablen. Die Buchstaben und Symbole müssen dabei, wie oben gezeigt, in Anführungszeichen stehen.

**3**

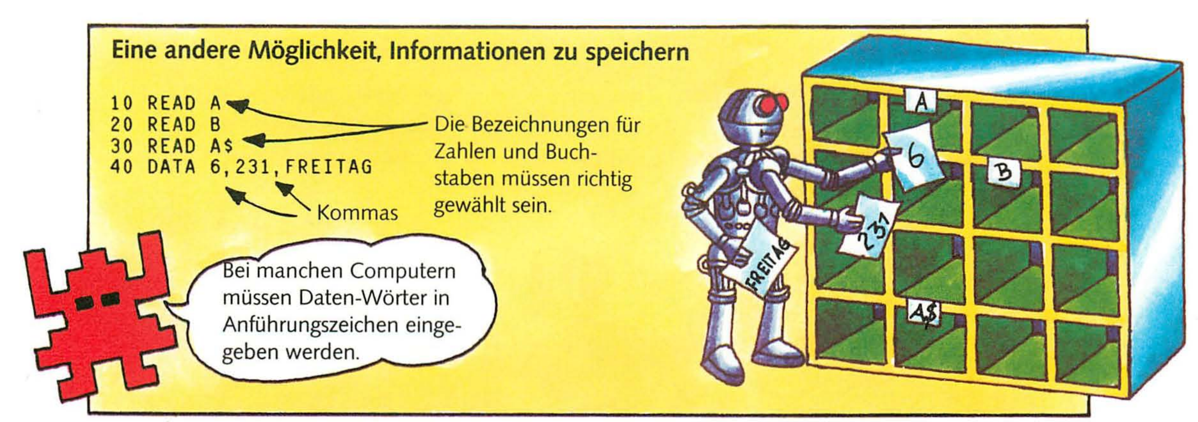
```
10 LET B=365
20 LET T$="TAGE IM JAHR"
30 LET L$="AUSSER IM SCHALTJAHR"
40 PRINT B
50 PRINT D$
60 PRINT L$
70 END
```

← Viele Computer benötigen END nicht.




Zur Angabe der Information auf dem Bildschirm wird das Wort PRINT zusammen mit dem Namen der Variablen eingegeben, z. B. PRINT A$. Das kurze Programm oben druckt die Information der Variablen B, D$ und L$ aus.

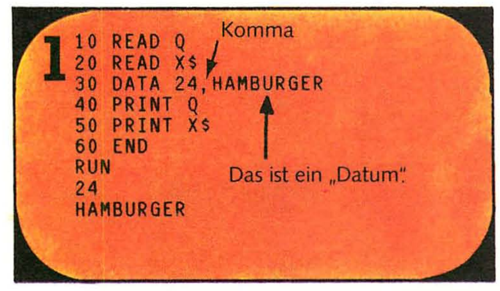
Man kann das Programm beliebig oft laufen lassen – der Computer druckt jedesmal die gleiche Information aus. Die Daten in den Variablen bleiben so lange gleich, bis man sie ändert.




Eine andere Möglichkeit der Informationsspeicherung besteht mit den Wörtern READ und DATA, wie oben gezeigt. Die READ-Zeilen bewirken, daß der Computer Speicherplatz benennt, die DATA-Zeile enthält die Information.

Wenn das Programm läuft, legt der Computer die Daten der Reihe nach in einem Speicherplatz ab. Die einzelnen Daten müssen durch Kommas voneinander getrennt sein, damit der Computer sie erkennen kann. (Diese Methode funktioniert nicht auf dem ZX81-Computer.)

## Programme




```
10 READ Q
20 READ X$
30 DATA 24,HAMBURGER
40 PRINT Q
50 PRINT X$
60 END
RUN
24
HAMBURGER
```

Hier sind zwei Programme. Eines verwendet READ und DATA, das andere LET, um Informationen im Speicher des Computers abzulegen.



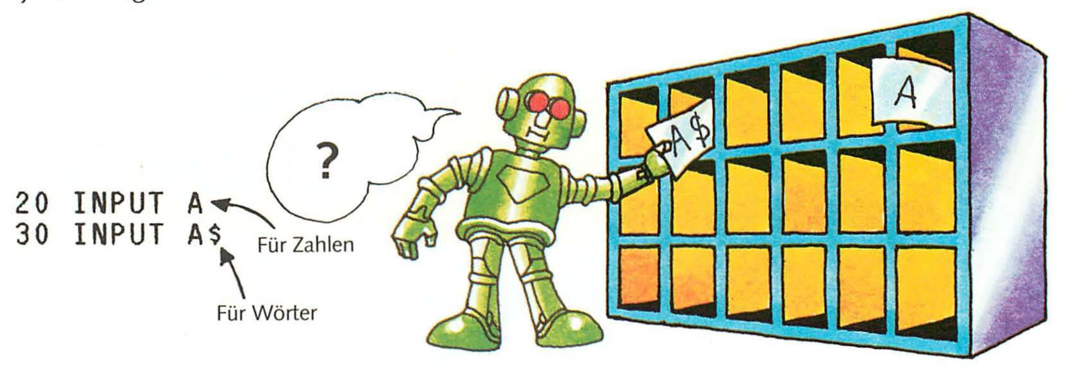

```
10 LET A$="ROBOTER SIND NÜTZLICH"
20 LET B$="SIE SIND"
30 LET C$="NÜTZLICHE BLECH-IDIOTEN"
40 PRINT A$
50 PRINT B$
60 PRINT C$
70 END
RUN
ROBOTER SIND NÜTZLICH
SIE SIND
NÜTZLICHE BLECH-IDIOTEN
```




Variablen sind benannte Plätze im Speicher des Computers, in denen Informationen gespeichert werden. Eine Variable, die Zahlen enthält, heißt *Zahlen-Variable*. Eine Variable, die Buchstaben oder Zeichen enthält, heißt *String-Variable*. Der Inhalt einer Variablen kann sich während des Programms ändern. Manche Computer verwenden Wörter als Bezeichnungen für Variablen, aber nicht solche Wörter, die in BASIC vorkommen; dies würde den Computer verwirren.

# INPUT

Eine weitere Möglichkeit, Daten in den Computer einzugeben, bietet das Wort INPUT. Es ermöglicht die Informationseingabe während des Programmablaufs. So kann man bei jedem Programmablauf verschiedene Daten benutzen.




INPUT verwendet man mit einem Namen, z. B. A für eine Zahl oder A$ für eine Zeichenkette. Stößt der Computer in einem Programm auf das Wort INPUT, belegt er einen Speicherplatz mit dem jeweiligen Namen und fragt nach den Daten – meist durch Angabe eines Fragezeichens oder eines anderen Symbols auf dem Bildschirm. Dann gibt man die Daten ein, der Computer speichert sie auf dem Speicherplatz und fährt mit dem restlichen Programm fort.




Bild 2 zeigt das Ergebnis dieses Programms. Sobald der Computer das Wort INPUT in Zeile 10 liest, gibt er ein Fragezeichen auf dem Bildschirm aus und wartet auf die Eingabe einer Zahl für G. Dann erscheint ein weiteres Fragezeichen im Anschluß an die INPUT-Anweisung in Zeile 20. Diesmal muß man Wörter oder Symbole eingeben; denn der Name B$ läßt den Computer eine Zeichenkette erwarten.




Wenn Sie einen Computer haben, versuchen Sie, dieses Programm einzugeben. Drücken Sie RUN, um das Programm zu starten. Erbittet der Computer Informationen, dann geben Sie Ihren Namen und Ihr Alter ein oder die entsprechenden Angaben eines Familienmitglieds. Lassen Sie dieses Programm öfter laufen, immer wieder mit anderen Daten. Um es zu starten, muß man jedesmal RUN eingeben. Der Computer gibt immer genau das aus, was man für N$ und A eingegeben hat.

## Verse schmieden mit dem Computer

Ihr BASIC reicht jetzt bereits, um mit dem Computer ein „Gedicht" zu schreiben. Hier ist ein Programm zum Schreiben von Gedichten, das PRINT und INPUT verwendet.

```
10 PRINT "WIE HEISSEN SIE?"
20 INPUT N$
30 PRINT "EIN GEDICHT VON"
40 PRINT N$
50 PRINT "NENNEN SIE EIN WORT, DAS"
60 PRINT "SICH AUF WISSEN REIMT"
70 INPUT A$
80 PRINT "HIER DAS GEDICHT:"
90 PRINT "VON COMPUTERN WOLLTE
         ICH NICHTS WISSEN"
100 PRINT "HEUTE MÖCHTE ICH SIE
          NICHT MEHR "
110 PRINT A$
120 END
```

Diese Zeile druckt Ihren Namen aus.

Diese Zeile druckt Ihr Wort aus.

```
RUN
WIE HEISSEN SIE?
? TOM
EIN GEDICHT VON TOM
NENNEN SIE EIN WORT, DAS
SICH AUF WISSEN REIMT
? MISSEN
HIER DAS GEDICHT:
VON COMPUTERN WOLLTE
ICH NICHTS WISSEN
HEUTE MÖCHTE ICH SIE
NICHT MEHR
MISSEN
```

Ihr Name und ein Wort

RUN drücken und noch einmal mit einem anderen Wort versuchen.

Das Programm bewirkt, daß der Computer die Antwort auf die Frage nach Ihrem Namen in N$ speichert und in Zeile 40 ausdruckt. Das von Ihnen gewählte Wort wird in A$ gespeichert und als Teil des Gedichts in Zeile 110 ausgedruckt. Wenn Sie einen Computer haben, lassen Sie das Programm mehrmals durchlaufen, jedesmal mit anderen Wort-Eingaben in Zeile 70.

### Programmierübung

Versuchen Sie ein Programm zu schreiben, das nach Ihrem Namen fragt, dann „Guten Tag" ausgibt, gefolgt von Ihrem Namen und einer Nachricht.

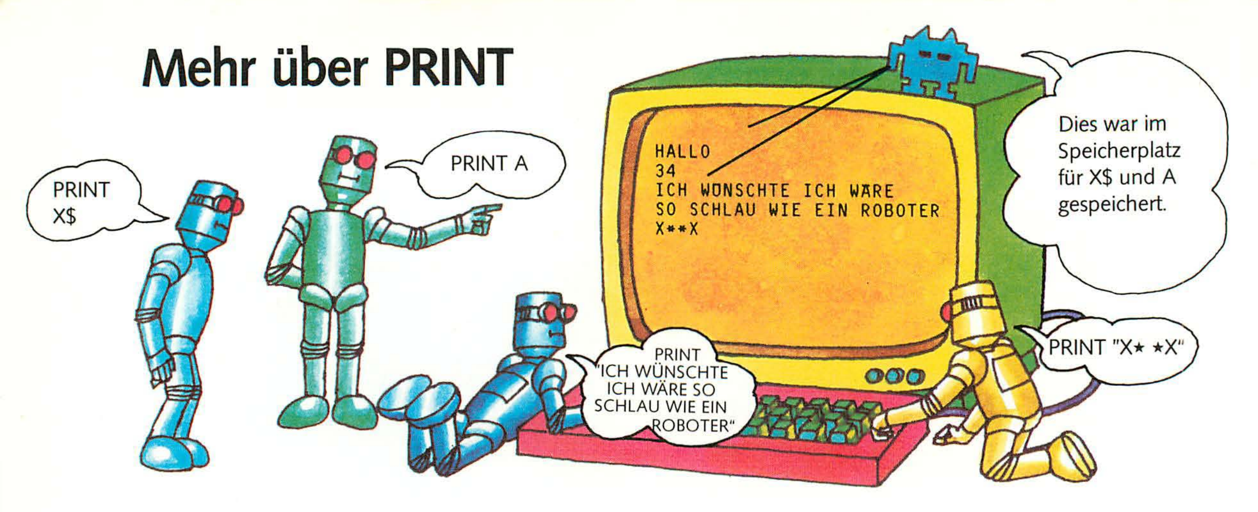
### Eingabe-Regeln für Programme

1. Ehe man ein neues Programm eingibt, drückt man NEW. Dies bewirkt, daß alle alten Programme und Variablen aus dem Speicher des Computers verschwinden.
2. Beim Eingeben des Programms nicht vergessen, am Ende jeder Zeile RETURN zu drücken (oder das entsprechende Wort für Ihren Computer)!
3. Nachdem das Programm eingegeben ist, sollten alle Zeilen auf dem Bildschirm nochmals auf Tippfehler überprüft werden. Prüfen Sie auch, ob keine Zeile fehlt.
4. Dann CLS (oder das entsprechende Wort für Ihren Computer) eingeben, um das Programm vom Bildschirm zu entfernen. Um das Programm zu starten, RUN eingeben.
5. Um das Programm nochmals zu überprüfen und um eventuell eine Zeile zu ändern, LIST eingeben. Um nochmals eine bestimmte Zeile zu überprüfen, kann man normalerweise LIST zusammen mit der Zeilennummer eingeben; dies sollte man jedoch prüfen, da diese Anweisung von Computer zu Computer etwas verschieden sein kann.
6. Die Unterbrechung eines laufenden Programms erfolgt mit BREAK oder ESCAPE. Suchen Sie den passenden Ausdruck in Ihrem Computer-Handbuch. Bei manchen Computern bewirkt ESCAPE das Löschen des gesamten Programms! Das Programm kann wieder gestartet werden, indem man RUN eingibt.

Hinweise, wie man Fehler findet, stehen auf Seite 42/43.

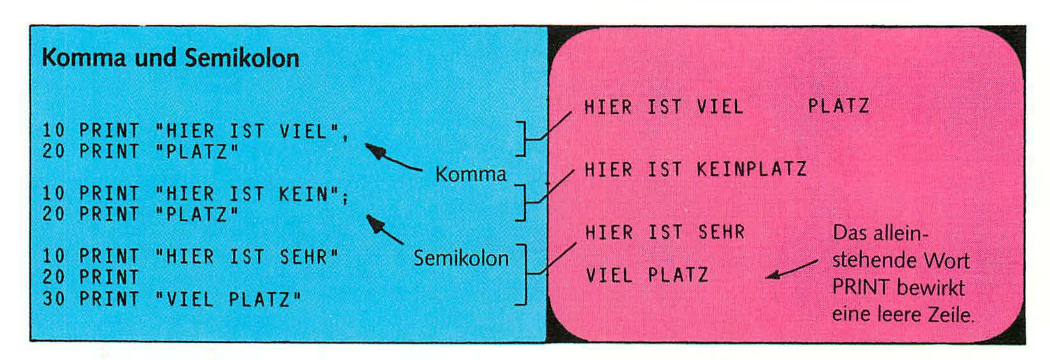


# Mehr über PRINT




Bisher haben wir gelernt, wie man PRINT zur Ausgabe von Wörtern und Zahlen auf dem Bildschirm und zur Ausgabe des Inhalts von Variablen verwendet. Im folgenden wird gezeigt, wie man Komma und Semikolon einsetzt, um Text auf dem Bildschirm anzuordnen. Man kann PRINT auch zur Ausführung von Rechnungen durch den Computer benutzen. Wie das gemacht wird, steht unten auf dieser Seite. Auf der nächsten Seite finden Sie weitere Einzelheiten zum Thema Variablen.

## Komma und Semikolon



```
10 PRINT "HIER IST VIEL",
20 PRINT "PLATZ"

10 PRINT "HIER IST KEIN";
20 PRINT "PLATZ"

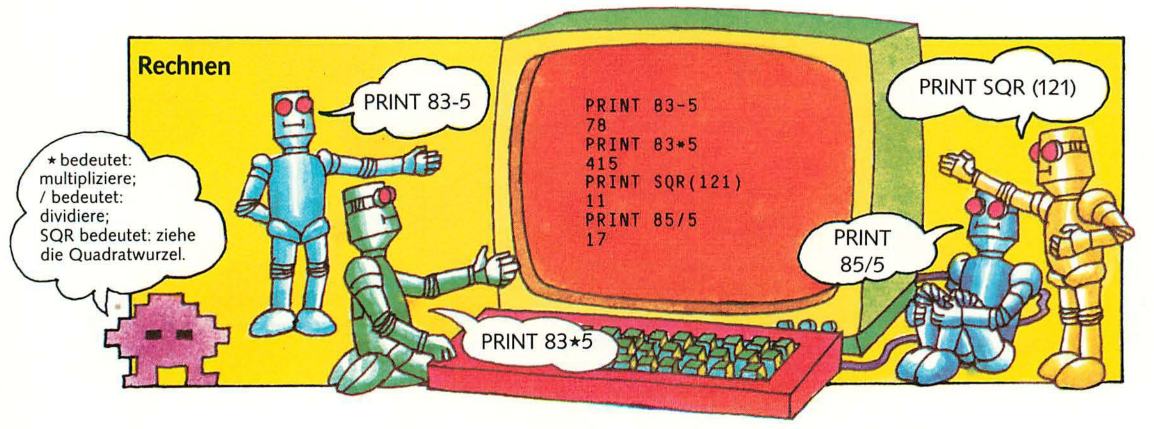
10 PRINT "HIER IST SEHR"
20 PRINT
30 PRINT "VIEL PLATZ"
```



Diese Zeilen zeigen, wie man Komma und Semikolon verwendet, um dem Computer zu sagen, wo er den nächsten Buchstaben setzen soll. Ein Komma läßt ihn etwas vorrücken, bei einem Semikolon bleibt er, wo er ist.

Das Bild oben zeigt die ausgegebenen Zeilen auf dem Bildschirm. Das alleinstehende Wort PRINT bewirkt eine leere Zeile.

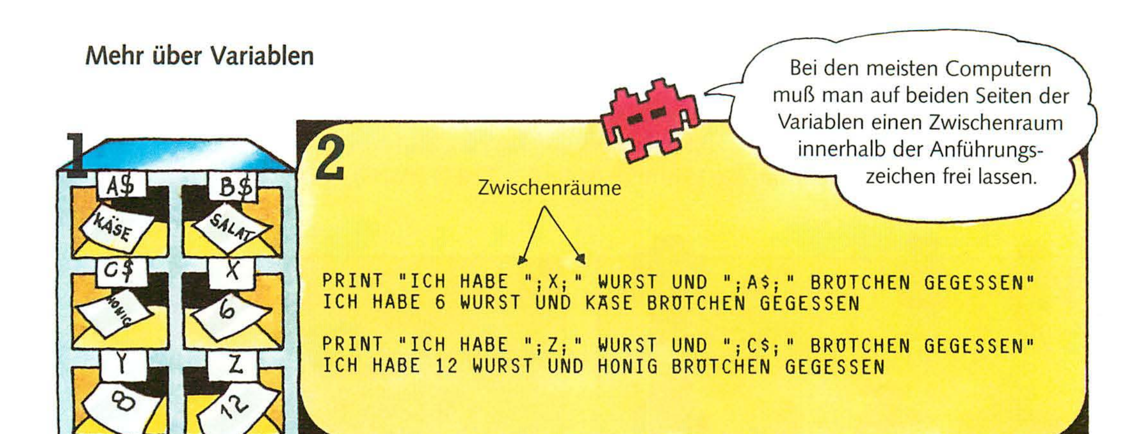
## Rechnen



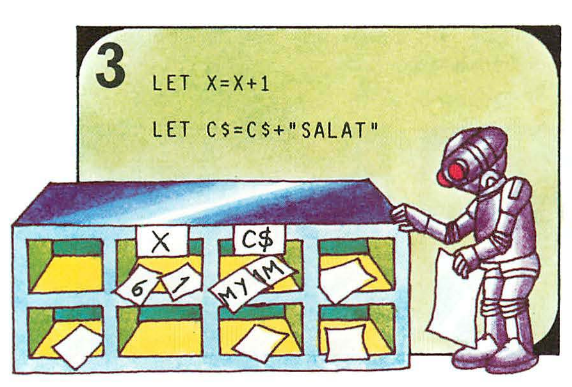

So wird PRINT angewendet, um Rechnungen auszuführen. Man benutzt dazu die üblichen Zeichen für Addition und Subtraktion, * für Multiplikation und / für Division.

Der Computer kann auch viel kompliziertere Rechenarbeiten ausführen: Sinus, Cosinus, Quadratwurzel usw.

## Mehr über Variablen



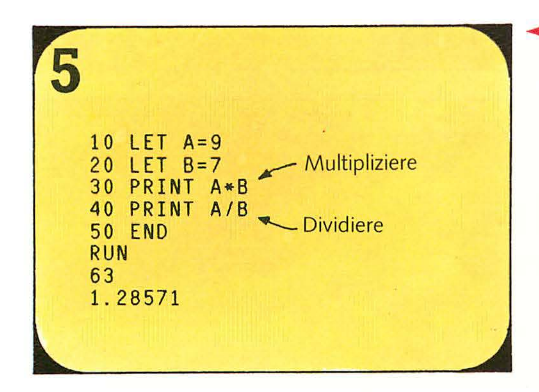

Nur die Variablen selbst auszudrucken, ist nicht sehr sinnvoll. Besser ist es, sie in einen Text einzufügen. Um Wörter und Variablen zusammen auszudrucken, müssen die Wörter wie gewöhnlich in Anführungszeichen stehen, und die Variable muß auf jeder Seite ein Semikolon haben, wie oben gezeigt. Wenn man die Information räumlich besser aufteilen will, kann man Kommas anstelle der Semikolons einsetzen.




Man kann in einem Programm den Inhalt der Speicherplätze verändern, wie im Bild gezeigt. Für den Computer bedeuten diese Anweisungen folgendes: Addiere 1 zu der Zahl im Speicherplatz X und füge „SALAT" zu den Buchstaben in C$ hinzu.




Bei wiederholtem Ausdruck der Variablen erscheinen die neuen Zahlen und Wörter, die jetzt in den Speicherplätzen gespeichert sind.




Man kann Variablen auch multiplizieren oder dividieren, wie im Programm oben gezeigt wird. Der Computer sucht die Zahlen in den Speicherplätzen und berechnet die Ergebnisse.

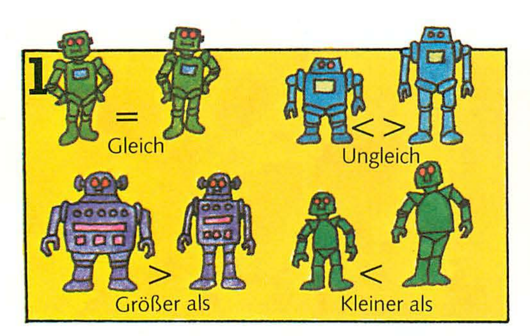
### Programmierübungen

1. Schreiben Sie ein Programm, das Zahlen zu den Variablen des Programms von Bild 5 addiert (bzw. von den Variablen abzieht), so daß die Resultate 100 und 1 in einer Zeile mit einem Zwischenraum ausgedruckt werden.
2. Ändern Sie die Zeilen 30 und 40 so, daß die Zahlen, die Rechenoperation und das Resultat ausgedruckt werden, z. B. "9 mal 3 gleich 63".
3. Ändern Sie das Programm in der Übung auf Seite 15 so, daß Ihr Name und die Nachricht in einer Zeile ausgedruckt werden.

# Wie der Computer Dinge vergleicht

Zu den nützlichsten Aufgaben, die ein Computer erledigen kann, gehört es, Informationen zu vergleichen und entsprechend den Ergebnissen zu handeln. Dazu verwendet man die Worte IF (wenn) ... THEN (dann).




Der Computer kann mit Hilfe verschiedener Tests Informationen miteinander vergleichen. Die Symbole für diese Tests sind oben zu sehen. Er kann testen, ob zwei verschiedene Daten gleich sind, verschieden sind oder ob ein Datum größer oder kleiner als das andere ist.



2

```
IF A=B THEN PRINT "BEIDE SIND GLEICH"
IF A>B THEN PRINT "A IST GRÖSSER"
IF A<B THEN PRINT "A IST KLEINER"
IF A<>B THEN PRINT "SIE SIND UNGLEICH"
```

Diese Zeilen zeigen, wie die Symbole mit IF und THEN benutzt werden, um den Computer zu veranlassen, zwei Daten miteinander zu vergleichen. Man kann beliebige Arten von Daten vergleichen: Wörter, Zahlen und Variablen, d. h. auch die Inhalte von Speicherplätzen.



3 **Wetter-Programm**

```
10 PRINT "WIE IST DAS WETTER HEUTE?"
20 INPUT W$
30 IF W$="REGEN" THEN PRINT "REGENSCHIRM"
40 IF W$="SONNE" THEN PRINT "BADEHOSE"
50 END
```





4

```
RUN
WIE IST DAS WETTER HEUTE?
? SONNE
BADEHOSE
RUN
WIE IST DAS WETTER HEUTE?
? REGEN
REGENSCHIRM
```

Dies ist ein Programm mit IF und THEN. In Zeile 20 speichert der Computer das Wort, das als Variable W$ eingegeben wurde. In den Zeilen 30 und 40 prüft er, ob das Wort in W$ entweder "Regen" oder "Sonne" heißt. Wenn ja, gibt er eine der vorgegebenen Antworten aus. Wird in Zeile 20 ein anderes Wort eingegeben, dann geschieht nichts. Man kann jedoch die Wörter in den Zeilen 30 und 40 ändern und dann versuchen, eines der neuen Wörter einzugeben.

5 **Alter-Programm**

```
10 PRINT "WIE ALT BIST DU?"
20 INPUT A
30 IF A>16 THEN PRINT "OPA"
40 IF A<16 THEN PRINT "BABY"
50 IF A=16 THEN PRINT "PARTNER"
RUN
WIE ALT BIST DU?
? 16
PARTNER
```

6 **Französisch-Lektion**

```
10 PRINT "WIE HEISST ROT IN FRANZÖSISCH?"
20 INPUT A$
30 IF A$="ROUGE" THEN PRINT "RICHTIG"
40 IF A$<>"ROUGE" THEN PRINT "FALSCH, ROUGE"
RUN
WIE HEISST ROT IN FRANZÖSISCH?
? BLEU
FALSCH, ROUGE
```

Im Alter-Programm vergleicht der Computer die Eingabe A mit der Zahl 16. Wenn sie größer ist als 16, druckt er "Opa". Ist sie kleiner als 16, druckt er "Baby"; ist sie 16, druckt er "Partner". Im anderen Programm druckt der Computer eine von zwei möglichen Antworten aus, je nachdem, ob A$ gleich "rouge" ist oder nicht.





**Programmierübung:** Schreiben Sie ein Programm, in dem der Computer Ihnen Rechenaufgaben stellt und dann entweder "richtig" oder die richtige Antwort ausgibt.

## Programmverzweigungen







Im Anschluß an das Wort THEN kann man dem Computer fast jede Anweisung geben. Eine nützliche Anweisung besteht z. B. darin, in eine andere Zeile zu springen. (Bei den meisten Computern, nicht jedoch beim ZX81, kann man das Wort GOTO weglassen.) In Programmen mit GOTO braucht man normalerweise eine STOP-Anweisung, sonst wiederholt der Computer das Programm ständig.

### Mathematik-Programm



```
10 PRINT "GIB EINE ZAHL EIN"
20 INPUT A
30 PRINT "GIB EINE WEITERE ZAHL EIN"
40 INPUT B
50 PRINT "MÖCHTEST DU ADDIEREN,"
60 PRINT "SUBTRAHIEREN, MULTIPLIZIEREN,"
65 PRINT "DIVIDIEREN ODER NICHTS TUN?"
70 INPUT C$
80 IF C$="ADDIEREN" THEN PRINT A+B
90 IF C$="SUBTRAHIEREN" THEN PRINT A-B
100 IF C$="MULTIPLIZIEREN" THEN PRINT A*B
110 IF C$="DIVIDIEREN" THEN PRINT A/B
120 IF C$="NICHTS TUN" THEN STOP
130 GOTO 10
```

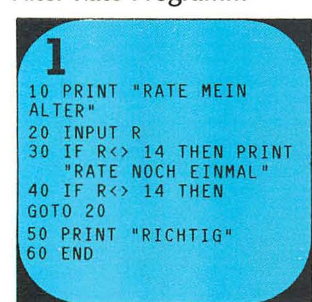
```
RUN
GIB EINE ZAHL EIN
? 17
GIB EINE WEITERE ZAHL EIN
? 184
MÖCHTEST DU ADDIEREN,
SUBTRAHIEREN, MULTIPLIZIEREN,
DIVIDIEREN ODER NICHTS TUN?
? ADDIEREN
201
GIB EINE ZAHL EIN
?
```

DAS PROGRAMM ENDET ERST DANN, WENN SIE STOP EINGEBEN

Antwort des Computers

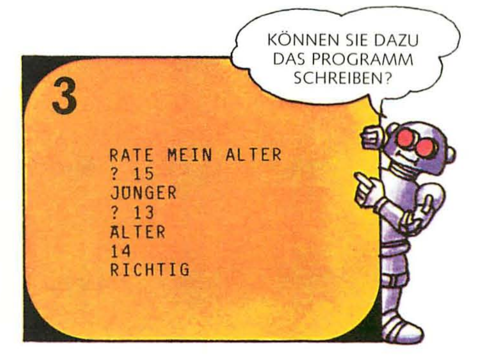
In diesem Programm werden die von Ihnen eingegebenen Zahlen A und B und Ihre Anweisungen in C$ gespeichert. In Zeile 80 bis 120 vergleicht der Computer C$ mit fünf verschiedenen Wörtern und führt die Anweisung aus, sobald er das richtige Wort gefunden hat. Er übergeht dabei die Zeilen, die nicht relevant sind.

### Alter-Rate-Programm










Dieses Programm läuft so oft, bis R=14 kommt. Wenn R=14 ist, übergeht der Computer die Zeilen 30 und 40 und druckt "richtig".

Können Sie das Programm so abändern, daß es Ihnen einige Hinweise gibt, so wie in Bild 3 gezeigt?

# Übungen mit BASIC

In den Programmen auf diesen beiden Seiten kommen die meisten der bisher behandelten BASIC-Ausdrücke vor. Das erste Programm ist ein Spiel für zwei Spieler mit dem Computer. Wenn Sie keinen Computer haben, studieren Sie die Programme und versuchen Sie zu verfolgen, wie sie funktionieren.



### Raumkommando

```
10 PRINT "GEGNER: QUADRAT NACH RECHTS:"
20 INPUT A
30 PRINT "GEGNER: QUADRAT NACH OBEN:"
40 INPUT B
50 CLS
60 PRINT "KOMMANDO: QUADRAT NACH RECHTS:"
70 INPUT C
80 PRINT "KOMMANDO: QUADRAT NACH OBEN:"
90 INPUT D
100 CLS
110 LET X=SQR((A-C)*(A-C)+(B-D)*(B-D))
120 PRINT "DU BIST JETZT"
130 PRINT X;"RAUMMEILEN ENTFERNT"
140 IF X<1.5 THEN PRINT "GEGNER AUFGESPÜRT"
150 IF X<1.5 THEN STOP
155 PRINT "BITTE NEUE POSITION EINGEBEN"
160 GOTO 10
170 END
```

Die Zeilen 10 bis 40 speichern die Koordinaten des Gegners in A und B.

Zeile 50 entfernt die Koordinaten des Gegners vom Bildschirm.

Die Zeilen 60 bis 90 speichern die Koordinaten des Kommandos in C und D.

Diese Zeile berechnet die Entfernung der beiden voneinander und speichert das Resultat in X.

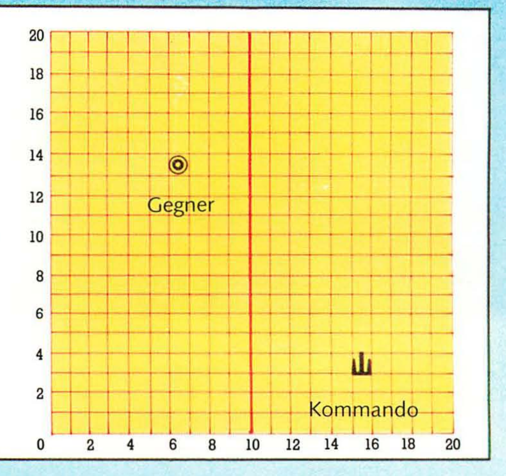
Ist X kleiner als 1,5, dann endet das Programm. Ist X größer als 1,5, dann läuft das Programm weiter.



In diesem Spiel ist der eine Spieler der Gegner und der andere das Raumkommando. Es versucht, den Gegner zu fangen. Jeder Spieler zeichnet für sich eine Karte und zeichnet dort seine Position ein. (Unten wird gezeigt, wie das gemacht wird.) Beide geben dem Computer die Gitterkoordinaten ihrer Position ein, und der Computer berechnet die Entfernung. Die Spieler benutzen die Ergebnisse des Computers, um die nächsten Schritte zu planen.

### Spielplan

Um zu einer Karte zu kommen, zeichnet jeder Spieler ein Gitter mit 20 x 20 Quadraten, die numeriert werden (siehe Bild rechts). Der Gegner startet auf der linken Seite des Gitters, das Kommando startet auf der rechten Seite. Bei jedem Zug kann sich der Spieler je zwei Quadrate nach oben, unten, zur Seite oder diagonal bewegen; dann gibt er dem Computer die jeweilige neue Position ein. Sind die Spieler weniger als 1,5 Raumeinheiten (d. h. Quadrate) voneinander entfernt, dann hat das Kommando den Gegner gefangen.



## Der schlaue Computer

In diesem Programm reagiert der Computer scheinbar klug auf die Beantwortung seiner Fragen. Wie das Programm läuft, steht unten auf dieser Seite. Das Programm verwendet INPUT in einer leicht veränderten Form, die es verkürzt und leichter lesbar macht.




Bei den meisten Computern (Ausnahme: ZX81) kann man die INPUT-Zeile übersichtlicher gestalten, indem man Wörter in Anführungszeichen vor den Variablen-Namen stellt.




Wenn das Programm läuft, erscheint das Eingabe-Fragezeichen hinter dem Text.

## Das Programm

```
5 LET C=0
10 PRINT "ICH MÖCHTE MICH MIT DIR UNTERHALTEN"
20 INPUT "ERZÄHLE MIR WAS DIR IN DIESER WOCHE
          PASSIERT IST";A$
30 READ B$
40 PRINT B$
50 INPUT C$
60 LET C=C+1
70 IF C=6 THEN GOTO 100
80 GOTO 30
90 DATA WARUM,WARUM DAS
95 DATA WIESO NUR,WARUM GENAU
98 DATA SAG MIR NUR WARUM, AUS WELCHEM GRUND
100 PRINT "ALSO LIEGT DER GRUND FÜR DEINE EINGABE"
110 PRINT "    ";A$
120 PRINT "IN WIRKLICHKEIT IN DEINER ANTWORT"
130 PRINT "    ";C$
140 PRINT "WIE SELTSAM!"
150 PRINT "LASS MICH NOCHMALS LAUFEN ZUR
          WEITEREN ERHELLUNG"
160 END
```

Dies ist die neue INPUT-Form. Ihre Antwort wird in A$ gespeichert.

In Zeile 30 sucht der Computer nach der ersten DATA-Zeile, nimmt das erste Textstück und speichert es in B$.

Zeile 40: Dies läßt den Computer in derselben Zeile bleiben.

Zeile 50: Ihre Antwort wird in C$ gespeichert.

Die Variable C in Zeile 60 und 70 ist ein Zähler, der zählt, wie oft das Programm wiederholt wird. Ist C = 6, sind alle DATA-Eingaben verbraucht, und der Computer springt zu Zeile 100.

Zeile 80 bewirkt Rücksprung zu Zeile 30, dabei werden die Daten in B$ durch den nächstfolgenden Text in der Datenliste ersetzt.

Die Leerstellen in Zeile 110 und 130 lassen freien Platz auf dem Bildschirm vor Ihren Antworten. Sie können beliebig viele Stellen freilassen.

## Programmablauf









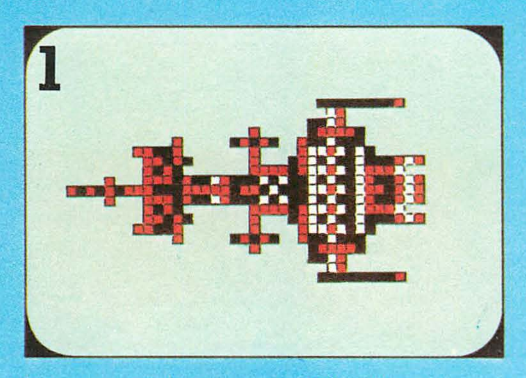




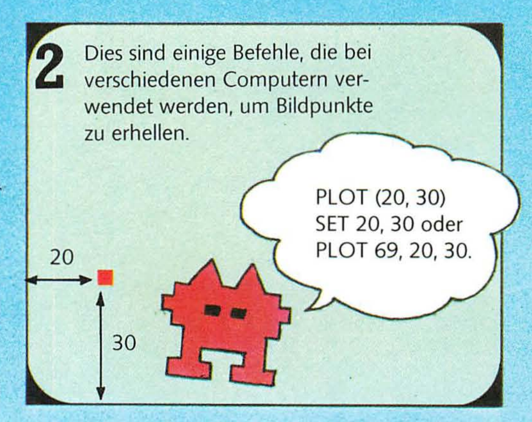


# Computer-Grafik

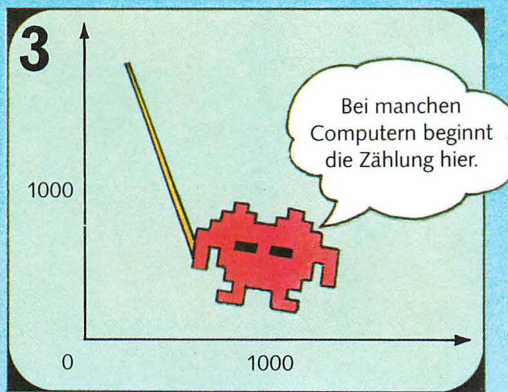
Der Computer kann Bilder „malen", indem kleine Flächen auf dem Bildschirm aufgehellt werden. Diese kleinen Flächen heißen *Bildpunkte*. Der Computer gibt jedem Bildpunkt eine entsprechende Anweisung, um ihn aufzuhellen. Die meisten Computer können die Bildpunkte auch mit verschiedenen Farben einfärben. Auf diesen beiden Seiten wird gezeigt, wie man BASIC anwendet, um einfache Bilder auf den Bildschirm zu bringen. Die hier gegebenen Anweisungen beziehen sich nur auf einfarbige Bilder.




Normalerweise kann man die Bildpunkte eines Computerbildes sehen. Computer mit großem Speicher können jedoch Bilder malen, die aus Tausenden winziger Bildpunkte bestehen. Diese Bilder heißen *hochauflösende Grafik*.

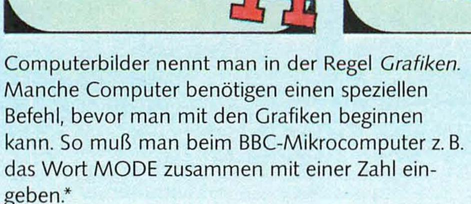



Diese Anweisung zur Aufhellung eines Bildpunktes ist von Computer zu Computer verschieden; normalerweise lautet sie PLOT (X, Y) oder ähnlich. X und Y sind die Koordinaten des Bildpunktes, d. h. X ist die Anzahl der Punkte nach rechts, und Y ist die Anzahl der Punkte nach oben.




Bei einem Computer mit hochauflösender Grafik kann man 1000 Punkte in der Breite und 1000 Punkte in der Höhe des Bildschirms zeichnen. Ein weniger leistungsfähiger Computer hat ungefähr 60 x 40 Bildpunkte. (Prüfen Sie die Angaben für Ihren Computer, damit Sie beim Zeichnen nicht außerhalb des Bildschirms geraten.)



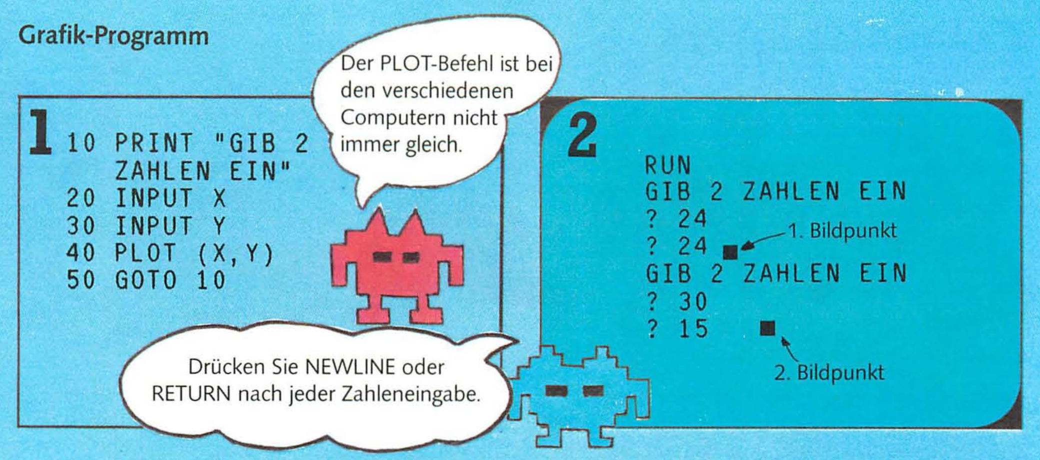
Computerbilder nennt man in der Regel *Grafiken*. Manche Computer benötigen einen speziellen Befehl, bevor man mit den Grafiken beginnen kann. So muß man beim BBC-Mikrocomputer z. B. das Wort MODE zusammen mit einer Zahl eingeben.*

Befehle wie UNPLOT (X, Y) schalten einen Bildpunkt aus. In den Programmen dieses Buches werden PLOT und UNPLOT verwendet. Wenn Sie einen Computer haben, prüfen Sie diese Befehle im Handbuch nach.



* Für die Programme in diesem Buch verwenden Sie MODE 5 auf dem BBC. Grafikbefehl: PLOT 69, X, Y, Löschbefehl: PLOT 71, X, Y.

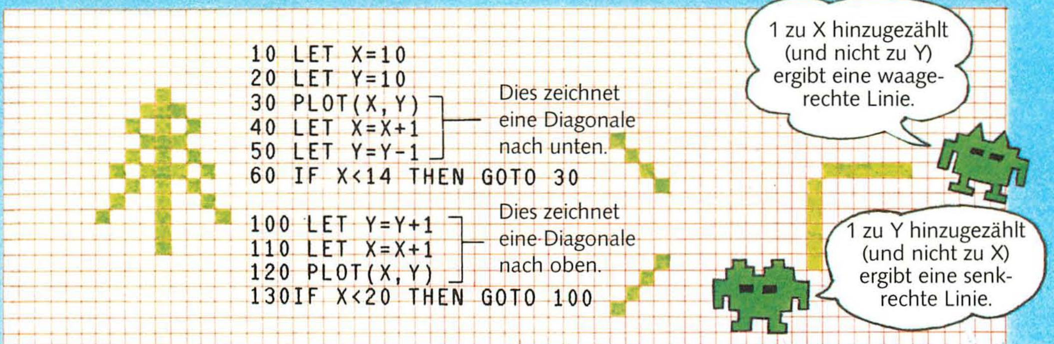
## Grafik-Programm




Dieses kurze Programm fragt nach zwei Zahlen und zeichnet dann den Bildpunkt mit diesen Zahlen als Koordinaten. Achten Sie beim Ausprobieren dieses Programms darauf, daß die eingegebenen Zahlen innerhalb des Zahlenbereichs Ihres Bildschirms liegen.

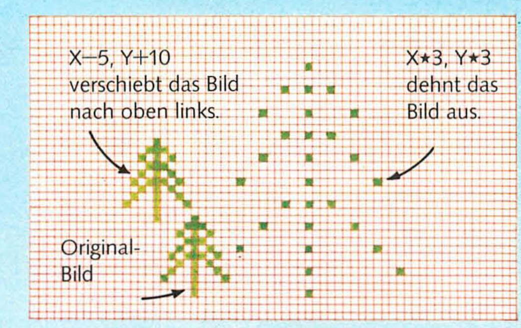
Zeile 50 bewirkt, daß das Programm ständig wiederholt wird; die einzige Möglichkeit, es zu beenden, besteht mit dem Befehl BREAK (oder dem entsprechenden Wort für Ihren Computer). Können Sie einen Zähler anbringen, der das Programm z. B. sechsmal laufen läßt (siehe Seite 21)?

### So malt man ein Bild mit dem Computer



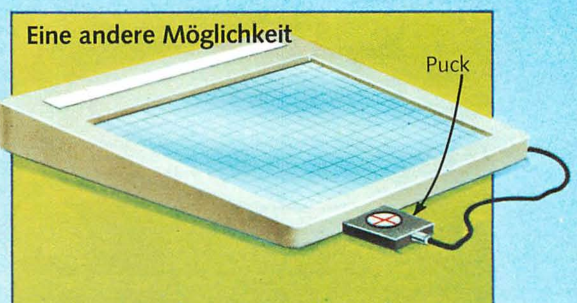

Zeichnen Sie das Bild zuerst auf kariertes Papier und berechnen Sie dann die Koordinaten der Quadrate.

Arbeiten Sie anschließend das vollständige Programm zum Zeichnen aller Quadrate aus. Geben Sie für X und Y Ausgangswerte an, addieren oder subtrahieren Sie diese und wiederholen Sie Teile des Programms. Damit erreichen Sie, daß der Computer Bildpunkte so zeichnet, wie es oben gezeigt wird.




Wenn das Programm einmal geschrieben ist, ist es leicht, das Bild durch Veränderung der Zahlen abzuändern. Man kann es auf einen anderen Platz auf dem Bildschirm verschieben, indem man die Anfangswerte ändert, oder man kann es vergrößern, indem man alle Zahlen mit 3 multipliziert.

### Eine andere Möglichkeit

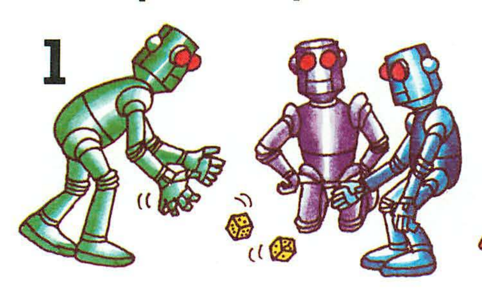



PLOT ermöglicht nur sehr einfache Bilder. Für kompliziertere Bilder braucht man spezielle Geräte, z. B. ein Grafiktablett. Auf das Tablett wird eine Zeichnung gemalt, die dann mit einem „Puck“ abgetastet wird. Auf diese Weise werden automatisch die Koordinaten in den Computer übertragen.



**Programmierübung:** Schreiben Sie ein Programm, das die Anfangsbuchstaben Ihres Namens auf den Bildschirm malt. Ein Beispiel dafür steht auf Seite 44.

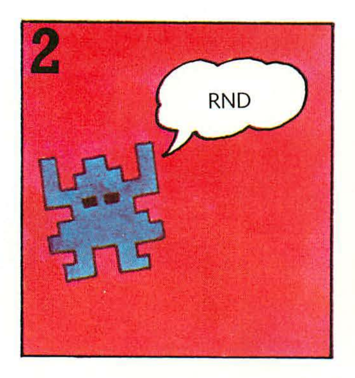
# Computer-Spiele

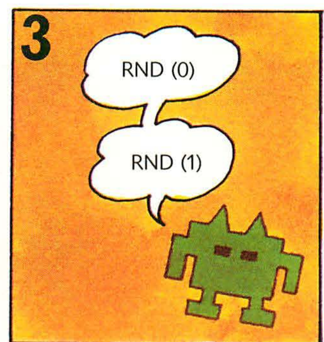


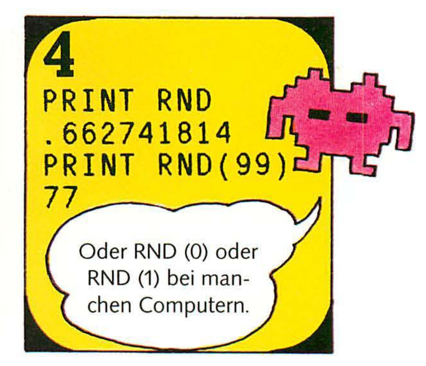



Beim Würfeln kann man die Ergebnisse nicht vorhersagen. Alle Zahlen zwischen 1 und 6 haben die gleiche Chance. Man kann unvorhersehbare Zahlen auch mit einem Computer herstellen. Solche Zahlen heißen *Zufallszahlen.*

Im Computer ist ein besonderes Programm für Zufallszahlen gespeichert. Manchmal wird eine bestimmte Zahl mehrere Male wiederholt. In den Folgen vieler Zufallszahlen erscheinen jedoch alle Zahlen ungefähr gleich oft.

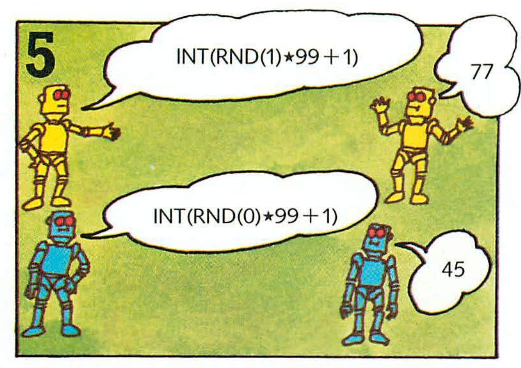


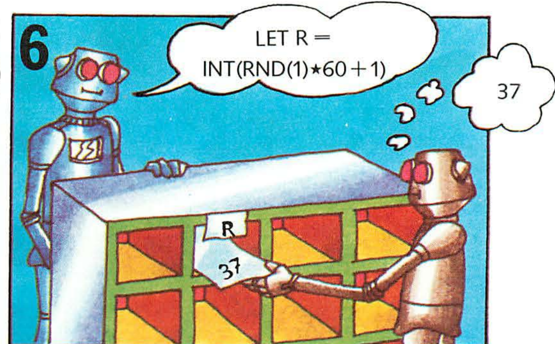




Wenn man RND eingibt, gibt der Computer eine Zufallszahl aus. Bei manchen Computern muß man im Anschluß an RND eine 1 oder 0 in Klammern eingeben. Wenn Sie einen Computer haben, überprüfen Sie das in Ihrem Handbuch.

Der RND-Befehl bewirkt die Angabe einer Zahl unter 1. Bei manchen Computern kann man RND und eine Zahl in Klammern angeben, z. B. RND (99). Dann wird eine ganze Zahl zwischen 1 und der Zahl in Klammern ausgegeben.




Bei manchen Computern muß man das Wort INT (Kurzform für engl. integer = ganze Zahl) eingeben, gefolgt von RND – entweder RND (1) oder RND (0). Dann multipliziert man mit der größten Zahl, die man haben möchte, und addiert 1. Auf diese Weise erhält man eine Zahl, die größer ist als 1.

Diese Anweisung bedeutet folgendes: Wähle eine Zufallszahl aus und speichere sie in der Variablen R. In diesem Buch bedeutet z. B. INT(RND(1)*60+1): Wähle eine Zufallszahl zwischen 1 und 60. Eventuell ist diese Anweisung für Ihren Computer (entsprechend den Angaben im Handbuch) etwas abzuändern.

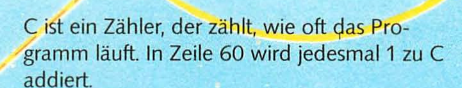


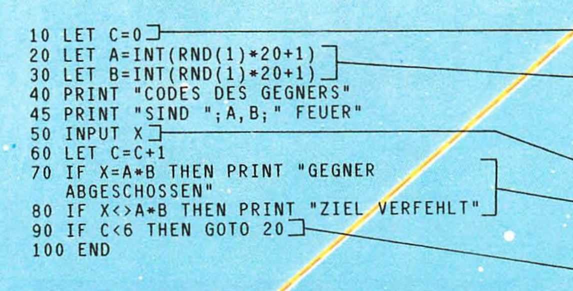


**Programmierübung:** Schreiben Sie ein Programm, das den Computer veranlaßt, eine Zufallszahl zwischen 10 und 20 auszugeben.

## Angriff im Weltraum

Dies ist ein Spielprogramm, in dem Zufallszahlen verwendet werden. Das Spiel geht so: Ihr Raumschiff wird von gegnerischen Fahrzeugen angegriffen. Der Computer des Raumschiffs ortet die Maschinen und gibt ihre codierten Positionen aus. Um die Fahrzeuge vom Bildschirm abzuräumen, muß man die Entfernung zu ihnen berechnen, indem man die Codes multipliziert und dann die Anwort eingibt.





```
10 LET C=0
20 LET A=INT(RND(1)*20+1)
30 LET B=INT(RND(1)*20+1)
40 PRINT "CODES DES GEGNERS"
45 PRINT "SIND ";A,B;" FEUER"
50 INPUT X
60 LET C=C+1
70 IF X=A*B THEN PRINT "GEGNER
   ABGESCHOSSEN"
80 IF X<>A*B THEN PRINT "ZIEL VERFEHLT"
90 IF C<6 THEN GOTO 20
100 END
```

C ist ein Zähler, der zählt, wie oft das Programm läuft. In Zeile 60 wird jedesmal 1 zu C addiert.

In diesen zwei Zeilen entstehen Zufallszahlen für die Codes des gegnerischen Fahrzeugs. Diese werden in A und B gespeichert.

Ihre Zahl wird in X gespeichert.

In Zeile 70 und 80 prüft der Computer, ob die Antwort richtig ist.

Diese Zeile bewirkt die Wiederholung des Programms, wenn C kleiner ist als 6.

## So läuft das Programm

Das rechte Bild zeigt den Ablauf des Programms. Wenn die richtige Antwort – Multiplikation der beiden Zahlen – eingegeben wird, dann gibt der Computer "Gegner abgeschossen" aus. Weicht die Antwort von AxB ab, dann gibt der Computer "Ziel verfehlt" aus.

```
RUN
CODES DES GEGNERS
SIND 17       3 FEUER
? 41
ZIEL VERFEHLT
CODES DES GEGNERS
SIND 11       5 FEUER
? 55
GEGNER ABGESCHOSSEN
CODES DES GEGNERS
SIND 13       6 FEUER
```

Das Komma in Zeile 45 bewirkt den Abstand der Zahlen voneinander.

## Programmierübung

Versuchen Sie, einen weiteren Zähler ins Programm einzubauen, der die Zahl der Treffer bestimmt und die Trefferpunkte am Ende des Spiels angibt. Dazu muß man eine Variable S mit dem Startwert 0 festlegen und bei jedem Treffer 1 addieren.

### Programm für Zufallsmuster

```
5 CLS
10 LET X=INT(RND(1)*30+1)
20 LET Y=INT(RND(1)*30+1)
30 PLOT (X,Y)
40 GOTO 10
```

Dadurch wird der Bildschirm freigemacht, ehe die Bildpunkte auf dem Bildschirm erscheinen.

Die Zufallszahlen müssen innerhalb des Bildschirms liegen.

Diese Zeile bewirkt die endlose Wiederholung des Programms.

Die Computer-Befehle für CLS, RND und PLOT können je nach Computer verschieden sein.

In diesem Programm werden Zufallszahlen verwendet, um Bildpunkte auf den Bildschirm zu zeichnen. In den Zeilen 10 und 20 werden Zufallszahlen zwischen 1 und 30 erzeugt und in X und Y gespeichert. Zeile 30 bewirkt das Zeichnen der Bildpunkte mit den Koordinaten X, Y. Während sich der Bildschirm mit Bildpunkten füllt, erscheinen immer weniger neue Punkte, weil viele bereits vorhanden sind. Um das Programm zu stoppen, gibt man BREAK oder ESCAPE ein bzw. das für Ihren Computer passende Wort.

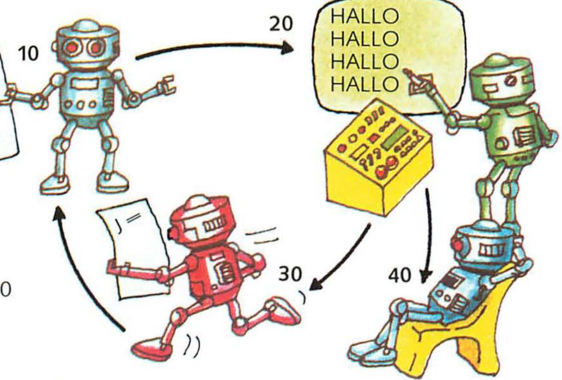
# Schleifen

Oft ist es in einem Programm erforderlich, daß der Computer einen bestimmten Vorgang mehrere Male wiederholt. Auf Seite 21 haben wir gesehen, wie solche Wiederholungen von Programmteilen mit GOTO und einer Variablen erfolgen, die als Zähler dient. Eine andere Möglichkeit besteht darin, dieselbe Zeile mehrmals zu wiederholen, indem die Befehle FOR... TO und NEXT verwendet werden. Dies nennt man *Schleife*.

## 1 Hallo-Schleife

```
10 FOR J=1 TO 6
20 PRINT "HALLO"
30 NEXT J
40 END
```

Schleife




Dieses Programm enthält eine Schleife von Zeile 10 bis 30. Sie bewirkt, daß der Computer Zeile 20 sechsmal wiederholt. Der Buchstabe J ist eine Variable. Zeile 10 veranlaßt den Computer, beim ersten Durchlauf J = 1 zu setzen, beim zweiten Durchlauf J = 2, dann 3 usw. bis 6. Zeile 20 bewirkt den Ausdruck des Wortes "Hallo". Zeile 30 läßt den Computer zurückgehen, um den nächsten Wert für J zu suchen. Sobald J = 6, springt der Computer zu Zeile 40.

## 2 Programm „Dumme Summe"




Schleife

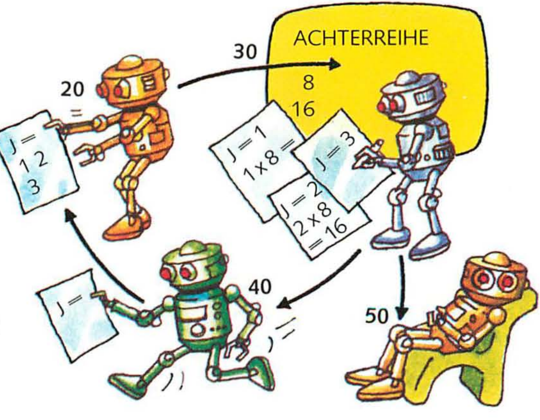
```
10 FOR J=1 TO 8
20 PRINT "2 PLUS 2 IST 5"
30 NEXT J
40 PRINT
50 PRINT "NUR EIN SCHERZ!"
60 END
```

In diesem Programm bewirkt die Schleife von Zeile 10 bis 30, daß der Computer Zeile 20 achtmal wiederholt. Jedesmal, wenn Zeile 20 an der Reihe ist, gibt der Computer die gleiche „dumme Summe" aus. Nachdem dies achtmal durchgeführt wurde, fährt der Computer mit dem restlichen Programm fort. Zeile 40 bewirkt lediglich eine leere Zeile.

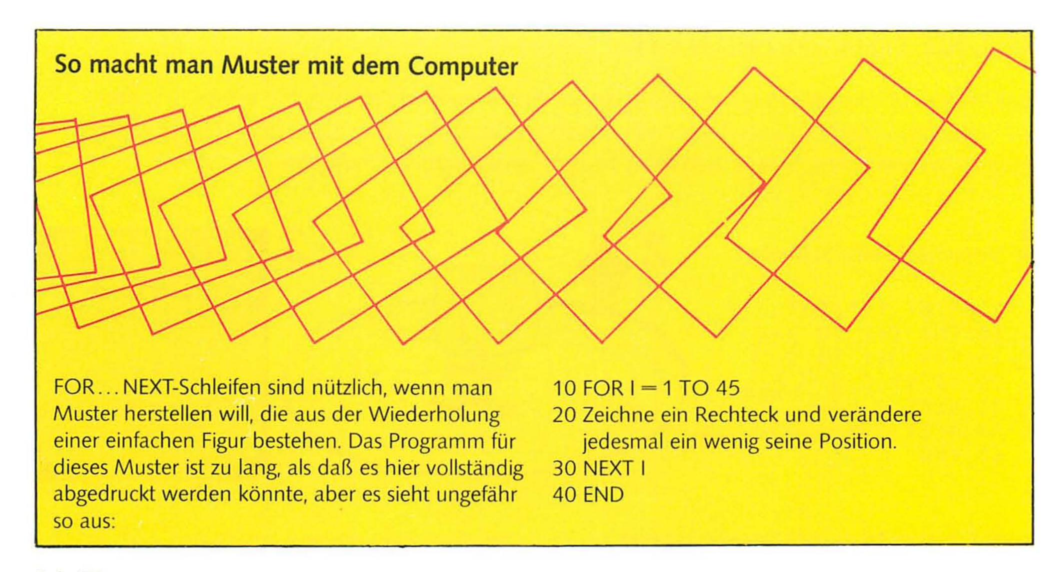
## 3 Achterreihen-Programm

```
10 PRINT "ACHTERREIHE"
20 FOR J= 1 TO 12
30 PRINT J*8
40 NEXT J
50 END
```

Schleife




In diesem Programm wird J sowohl als Schleifenzähler als auch als Teil der Multiplikation J*8 verwendet. Zeile 20 gibt die Anweisung, zuerst J = 1, dann 2, 3 usw. bis 12 zu setzen. Zeile 30 nimmt den gerade bestehenden Wert von J, multipliziert ihn mit 8 und gibt die Antwort aus. Zeile 40 bewirkt einen Rücksprung auf Zeile 20 zur Suche des nächsten Wertes für J.



## So macht man Muster mit dem Computer

FOR... NEXT-Schleifen sind nützlich, wenn man Muster herstellen will, die aus der Wiederholung einer einfachen Figur bestehen. Das Programm für dieses Muster ist zu lang, als daß es hier vollständig abgedruckt werden könnte, aber es sieht ungefähr so aus:

```
10 FOR I = 1 TO 45
20 Zeichne ein Rechteck und verändere
   jedesmal ein wenig seine Position.
30 NEXT I
40 END
```

## Schritte

Manchmal ist es nützlich, den Wert von J in anderen als Einer-Schritten zu ändern. So möchte man z. B. in Dreier-Schritten nach oben oder lieber in Siebener-Schritten nach unten gehen. Dazu dient das Wort STEP (engl. step = Schritt). Im folgenden Programm bewirkt STEP –1, daß jedesmal, wenn der Computer die Schleife in Zeile 10 bis 40 abarbeitet, J um 1 kleiner wird.

## Programm "Gefräßiger Computer"

Die Zahl 2 stoppt die Schleife nach J = 2, d. h. wenn noch ein Keks da ist.

```
5 CLS
10 FOR J=7 TO 2 STEP -1
20 PRINT "ES SIND NOCH ";J;" KEKSE DA"
30 NEXT J
40 PRINT                        Schleife
50 PRINT "ICH PLATZE"
60 FOR K=1 TO 1000
70 REM VERZÖGERUNG
80 NEXT K                       Schleife
90 PRINT
100 PRINT "*W*U*M*M*"
```



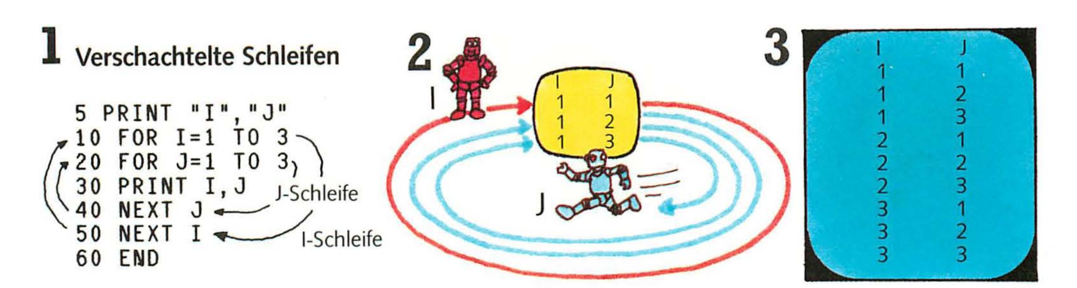

In diesem Programm sind zwei Schleifen. Die Schleife von Zeile 10 bis 30 bewirkt, daß Zeile 20 sechsmal ausgedruckt wird. Jedesmal wird der Wert von J um 1 vermindert, und der entsprechende Wert von J wird in Zeile 20 ausgegeben. Bei der Schleife von Zeile 60 bis 80 tut der Computer nichts. Er durchläuft lediglich alle Werte für K von 1 bis 1000, dadurch entsteht eine kleine Pause. Zeilen, die mit REM (Kurzform für engl. remark = Bemerkung) beginnen, ignoriert der Computer. REM-Zeilen dienen dazu festzuhalten, was das Programm macht.

## Programmierübungen

1. Ändern Sie das Programm "Achterreihe" von Seite 26 so, daß es "1x8=" gefolgt vom Ergebnis ausdruckt.
2. Versuchen Sie ein Programm zu schreiben für das „1 mal N", d. h. ein Programm, das für jede beliebige Zahl N, die eingegeben wird, die entsprechende 1x1-Reihe ausgibt. Hinweis: Zunächst läßt man den Computer nach einer Zahl N fragen. Dann benutzt man eine Schleife, um die 1x1-Reihe zu berechnen und die Ergebnisse auszugeben. Am Ende des Programms könnte man noch einige Zeilen hinzufügen, die abfragen, ob die 1x1-Reihe für eine andere Zahl gewünscht wird; dann kann man das Programm erneut laufen lassen.

# Tricks mit Schleifen

Hier finden Sie weitere Programme, in denen Schleifen vorkommen. Sie erfahren dabei, wie Schleifen innerhalb von Schleifen verwendet werden, um verschiedene Dinge gleichzeitig zu wiederholen. Solche Schleifen heißen *verschachtelte Schleifen.*

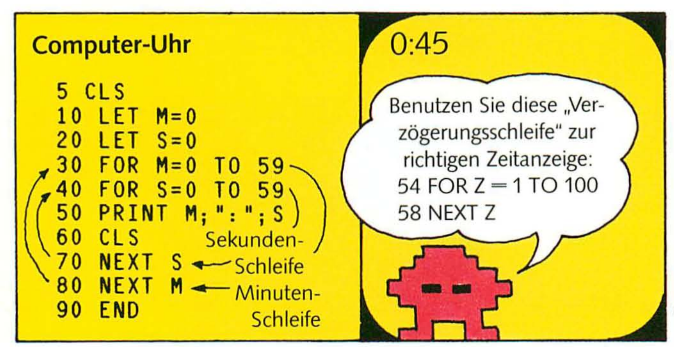



## 1 Verschachtelte Schleifen

```
5 PRINT "I","J"
10 FOR I=1 TO 3
20 FOR J=1 TO 3
30 PRINT I,J
40 NEXT J
50 NEXT I
60 END
```

```
I    J
1    1
1    2
1    3
2    1
2    2
2    3
3    1
3    2
3    3
```

Dieses Programm hat eine I-Schleife und eine J-Schleife. Die J-Schleife befindet sich innerhalb der I-Schleife. Jedesmal, wenn die I-Schleife bearbeitet wird, wird die J-Schleife dreimal wiederholt, wobei der Wert von J jedesmal ausgegeben wird. Das Bild oben zeigt das Ergebnis dieses Programms. Die Kommas bewirken, daß die Zahlen übersichtlich angeordnet sind.




### Computer-Uhr

```
5 CLS
10 LET M=0
20 LET S=0
30 FOR M=0 TO 59
40 FOR S=0 TO 59
50 PRINT M;":";S
60 CLS
70 NEXT S
80 NEXT M
90 END
```




### Fehler in Schleifen

```
10 FOR I=1 TO 4
20 FOR J=1 TO 4
30 PRINT I
40 PRINT J
50 NEXT I
60 NEXT J
```

Im Inneren eines Computers befindet sich eine elektronische „Uhr", die den Rhythmus für die gesamte Arbeit des Computers bestimmt. Die Uhr „tickt" zwischen ein- und viermillionenmal in der Sekunde. Das Programm oben bewirkt, daß der Computer sich wie eine Digitaluhr verhält. Das Programm enthält verschachtelte Schleifen: eine zählt die Sekunden, eine andere zählt die Minuten.

Die Sekundenschleife wird innerhalb jeder Minutenschleife 60mal ausgeführt. Wenn man dieses Programm auf einem Computer ausprobiert, wird es zunächst zu schnell ablaufen. Daher ist es ratsam, eine spezielle „Verzögerungsschleife" einzubauen. Damit erreicht man, daß die Computeruhr genauso schnell „tickt" wie eine richtige Uhr.

### Zufallszahlen-Test

```
10 FOR I=1 TO 1000
20 LET R=INT(RND(1)*6+1)
30 IF R=1 THEN LET A=A+1
40 IF R=2 THEN LET B=B+1
50 IF R=3 THEN LET C=C+1
60 IF R=4 THEN LET D=D+1
70 IF R=5 THEN LET E=E+1
80 IF R=6 THEN LET F=F+1
90 NEXT I
100 PRINT "ENDE"
110 PRINT A,B,C
120 PRINT D,E,F
130 END
```

```
RUN
ENDE
162       168       167
160       187       156
```

Dieses Programm dauert sehr, sehr lange. Man kann es verkürzen, indem man die Zahl in Zeile 10 ändert: z. B. in 500 oder 250.

Dieses Programm zeigt an, ob RND wirklich funktioniert. Die Schleife von Zeile 10 bis 90 bewirkt, daß der Computer tausendmal eine Zufallszahl zwischen 1 und 6 auswählt. Dabei wird gezählt, wie oft jede Zahl in den Variablen A bis F genannt wird. Dann werden die Ergebnisse ausgedruckt.*

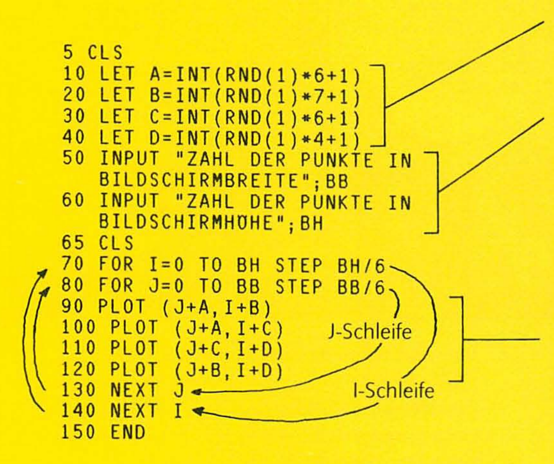


* Bei manchen Computern, z. B. ZX81 und BBC, sind zu Beginn des Programms zusätzliche Zeilen erforderlich, um jede Variable auf 0 zu setzen.

## Programm "Muster wiederholen"

Dieses Programm verwendet verschachtelte Schleifen, um ein kleines Muster auf dem Bildschirm zu wiederholen. Das Programm sieht ziemlich kompliziert aus; wenn man es jedoch sorgfältig durchliest und überlegt, was in jeder Zeile geschieht, erkennt man rasch, wie es funktioniert. Die Form des Musters wird durch Zufallszahlen bestimmt, sie verändert sich daher bei jedem Programmdurchlauf.

Bei Computern mit hochauflösender Grafik kann man höhere Zufallszahlen verwenden.



```
5 CLS
10 LET A=INT(RND(1)*6+1)
20 LET B=INT(RND(1)*7+1)
30 LET C=INT(RND(1)*6+1)
40 LET D=INT(RND(1)*4+1)
50 INPUT "ZAHL DER PUNKTE IN
   BILDSCHIRMBREITE";BB
60 INPUT "ZAHL DER PUNKTE IN
   BILDSCHIRMHÖHE";BH
65 CLS
70 FOR I=0 TO BH STEP BH/6
80 FOR J=0 TO BB STEP BB/6
90 PLOT (J+A,I+B)
100 PLOT (J+C,I+C)
110 PLOT (J+C,I+D)
120 PLOT (J+B,I+D)
130 NEXT J
140 NEXT I
150 END
```



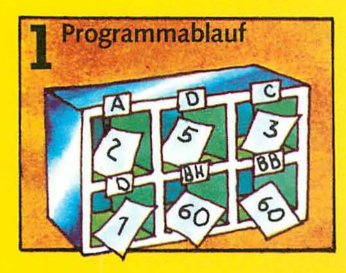
Diese Zeilen bestimmen die Zufallszahlen für das Muster und speichern sie in A, B, C und D.

Zeile 50 und 60 erfragen Breite (BB) und Höhe (BH) Ihres Bildschirms.

Die I-Schleife zählt, wie oft das Muster auf dem Bildschirm nach oben erscheint. I wächst jedesmal um die Höhe des Bildschirms (BH) geteilt durch 6; das Muster erscheint daher sechsmal in der Höhe des Bildschirms.

Bei jeder Wiederholung der Schleifen veranlassen die Zeilen 90 bis 120 den Computer, vier Bildpunkte zu zeichnen mit den gerade geltenden Werten für I und J plus Zufallszahlen.

Die J-Schleife zählt, wie oft das Muster auf dem Bildschirm nach rechts erscheint. Diese Schleife arbeitet genauso wie die I-Schleife.





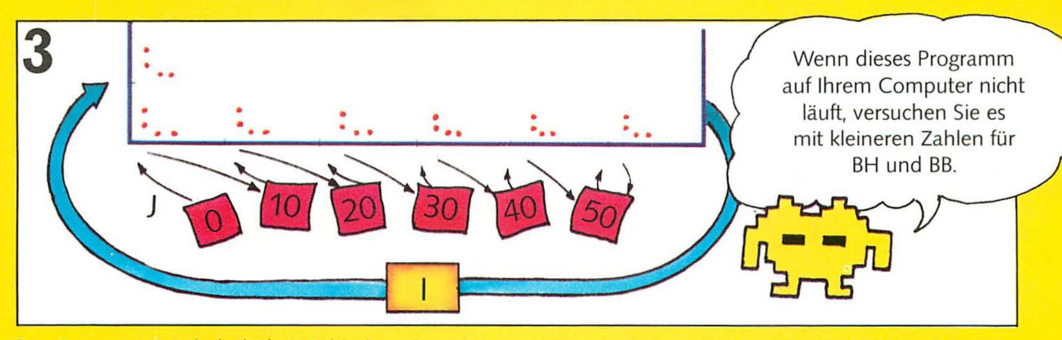
Wir nehmen einmal an, der Computer hat die Zufallszahlen 2, 5, 3 und 1 gewählt, und Breite und Höhe des Bildschirms sind jeweils 60.

**2**

2,5 — 12,5 — Erstes Muster — Zweites Muster — 2,3 — 12,3 — 3,1 — 5,1 — 13,1 — 15,1

0 2 4 6 8 10 12 14 16 18 20

Beim ersten Programmdurchlauf sind I und J gleich 0; der Computer benutzt also nur die Zufallszahlen, um das erste Punktmuster zu zeichnen. In Zeile 130 erfolgt ein Rücksprung, um den nächsten Wert für J zu bestimmen: J+60/6, d. h. 10. Dann wird das zweite Muster gezeichnet, und zwar mit J gleich Zufallszahlen plus 10. Auf diese Weise wiederholt sich das Muster in der Breite des Bildschirms.





Der Computer wiederholt die J-Schleife sechsmal, er addiert jedesmal 10 zu J hinzu und zeichnet so das Muster in der Breite des Bildschirms. Dann erfolgt ein Rücksprung, um den Wert für I zu suchen; dieser ist gleich 10. J wird wieder gleich 0 gesetzt, und der Computer zeichnet die nächste Muster-Zeile mit I gleich 10 und J, das jedesmal um 10 wächst.





**Programmierübung:** Versuchen Sie ein Wiederholungsprogramm zu schreiben, das ein Raumschiff-Muster auf dem Bildschirm erscheinen läßt. Auf Seite 45 finden Sie einige Hinweise dazu.

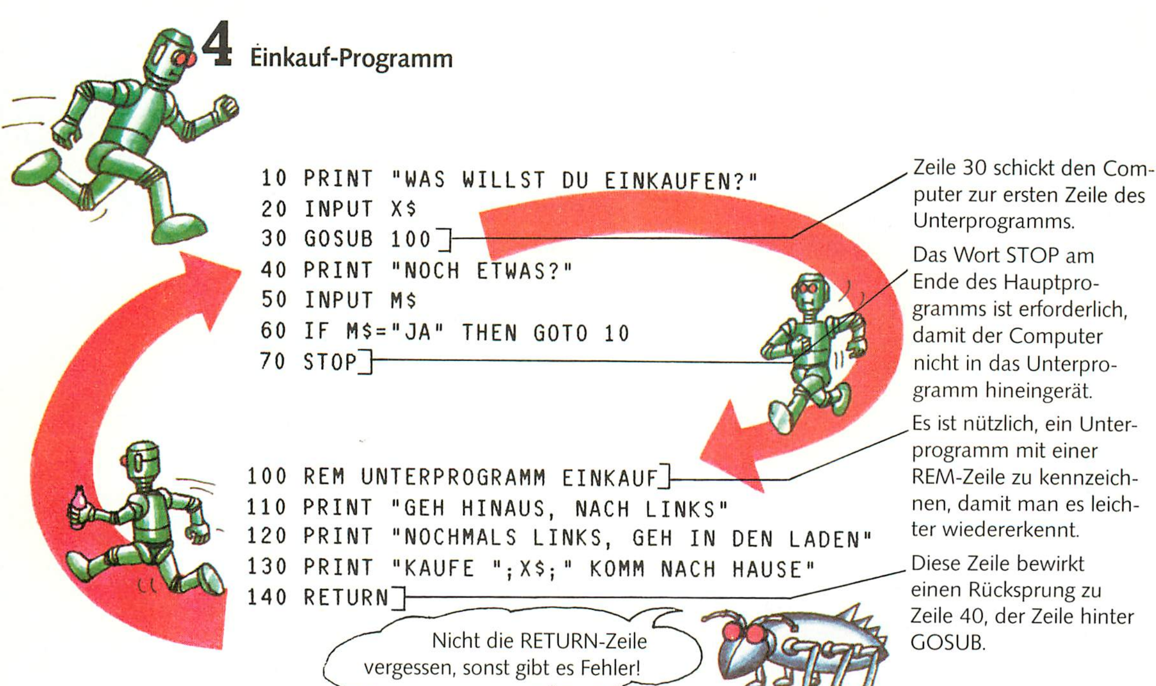
# Unterprogramme

Ein Unterprogramm (engl. subroutine) ist eine Art Mini-Programm innerhalb eines Programms. Man führt damit eine spezielle Aufgabe aus, z. B. die Addition von Zahlen, Festhalten von Punktzahlen usw. Der Computer kann jederzeit veranlaßt werden, ein solches Unterprogramm auszuführen. Man erspart sich dadurch, diesen Programmteil jedesmal Zeile für Zeile auszuschreiben. Das gesamte Programm wird dadurch kürzer und leichter lesbar, und man kann es schneller in den Computer eingeben.




Angenommen, Sie hätten einen Roboter, den Sie so programmieren könnten, daß er Besorgungen erledigt. Wenn er einkaufen soll, müßten Sie ihm genau sagen, wie er zum Laden kommt.

Jedesmal wenn der Roboter etwas kaufen soll, müßten Sie ihm die gleichen Anweisungen geben. Es wäre viel einfacher, dem Roboter ein Einkauf-Unterprogramm zu geben und ihn zu veranlassen, sich gegebenenfalls darauf zu beziehen.



**4 Einkauf-Programm**

```
10 PRINT "WAS WILLST DU EINKAUFEN?"
20 INPUT X$
30 GOSUB 100
40 PRINT "NOCH ETWAS?"
50 INPUT M$
60 IF M$="JA" THEN GOTO 10
70 STOP

100 REM UNTERPROGRAMM EINKAUF
110 PRINT "GEH HINAUS, NACH LINKS"
120 PRINT "NOCHMALS LINKS, GEH IN DEN LADEN"
130 PRINT "KAUFE ";X$;" KOMM NACH HAUSE"
140 RETURN
```

Zeile 30 schickt den Computer zur ersten Zeile des Unterprogramms.

Das Wort STOP am Ende des Hauptprogramms ist erforderlich, damit der Computer nicht in das Unterprogramm hineingerät.

Es ist nützlich, ein Unterprogramm mit einer REM-Zeile zu kennzeichnen, damit man es leichter wiedererkennt.

Diese Zeile bewirkt einen Rücksprung zu Zeile 40, der Zeile hinter GOSUB.



In BASIC wird das Wort GOSUB verwendet, wenn der Computer ein Unterprogramm abarbeiten soll. Am Ende des Unterprogramms steht das Wort RETURN. Im Anschluß an GOSUB sollte die Nummer der ersten Zeile des Unterprogramms stehen. (RETURN braucht keine Zeilennummer.) Der Computer springt automatisch zum Hauptprogramm zurück und bearbeitet die Stelle, die an der Reihe gewesen wäre, bevor er das Hauptprogramm verlassen hat. Ein Unterprogramm kann an jeder Stelle des Programms aufgerufen werden, sooft es gewünscht wird.

## GOSUB-Programme

Ein Unterprogramm ist dann nützlich, wenn eine bestimmte Aufgabe an verschiedenen Stellen des Programms öfter wiederholt werden soll. Hier sind noch weitere Programme mit Unterprogrammen.

### Zahlen-Programm

```
50 INPUT A
60 INPUT B
70 GOSUB 250
80 PRINT "A GETEILT DURCH B=";A/B
90 GOTO 50

250 REM UNTERPROGRAMM STOP
260 IF A=0 AND B=0 THEN STOP
270 RETURN
```

Dieses Unterprogramm ermöglicht es, aus dem Programm herauszugehen. Wenn der Divisionsvorgang gestoppt werden soll, gibt man 0 in den Zeilen 50 und 60 ein. Das Wort STOP braucht in diesem Programm nicht vor dem Unterprogramm zu stehen, da Zeile 90 für einen Rücksprung sorgt.

### Umwandlungs-Programm

```
100 INPUT "ENTFERNUNG";KM
110 INPUT "ZEIT";H
120 GOSUB 200
130 PRINT "DURSCHNITTSGESCHWINDIGKEIT"
140 PRINT KM/H;" KM/H UND";M/H;" M/H"
150 STOP
200 REM        UMWANDLUNG KM IN MEILEN
210 LET M=KM/1.609
220 RETURN
```

Dieses Unterprogramm verwandelt Kilometer in Meilen. Man kann ein Unterprogramm in vielen verschiedenen Programmen verwenden, muß allerdings darauf achten, daß immer die gleichen Variablen-Namen benutzt werden.

### Kreis-Programm

```
1 MITTELPUNKT = X,Y
2 RADIUS = R
3 FARBE = X
4 GOSUB 10
5 GOTO 1
10 REM        KREIS ZEICHNEN
11 ZEICHNE KREIS MIT MITTEL-
   PUNKT X,Y;RADIUS R; FARBE X
12 RETURN
```




Unterprogramme sind für Grafik-Programme nützlich, wenn Diagramme gezeichnet werden sollen mit Zahlen, die im Hauptprogramm errechnet worden sind. Mit diesem Programm könnte man viele verschiedene Kreise zeichnen, wenn man in den Zeilen 1 bis 5 entsprechende Informationen eingibt.

### Quiz-Programm

```
5 LET C=0
10 PRINT "WANN WURDE ERFUNDEN?"
20 READ C$,F
30 PRINT C$
40 INPUT A
50 LET C=C+1
60 IF C=3 THEN STOP
70 GOSUB 100
80 GOTO 10

100 REM ANTWORTEN
110 IF ABS(A-F)<10 THEN PRINT "STIMMT"
120 IF ABS(A-F)>10 THEN PRINT "FALSCH"
130 PRINT "NÄCHSTER VERSUCH"
140 RETURN

200 DATA TELEFON,1876,DRUCKERPRESSE,1450,FAHRRAD,1791
```

In Zeile 20 sucht der Computer die Datenzeile und speichert das erste Wort in C$ und die erste Zahl in F.

Ausdruck des Wortes aus C$.

Der Zähler C stoppt das Programm, wenn es dreimal gelaufen ist, weil sich nur drei Datenwörter in C$ und F befinden.

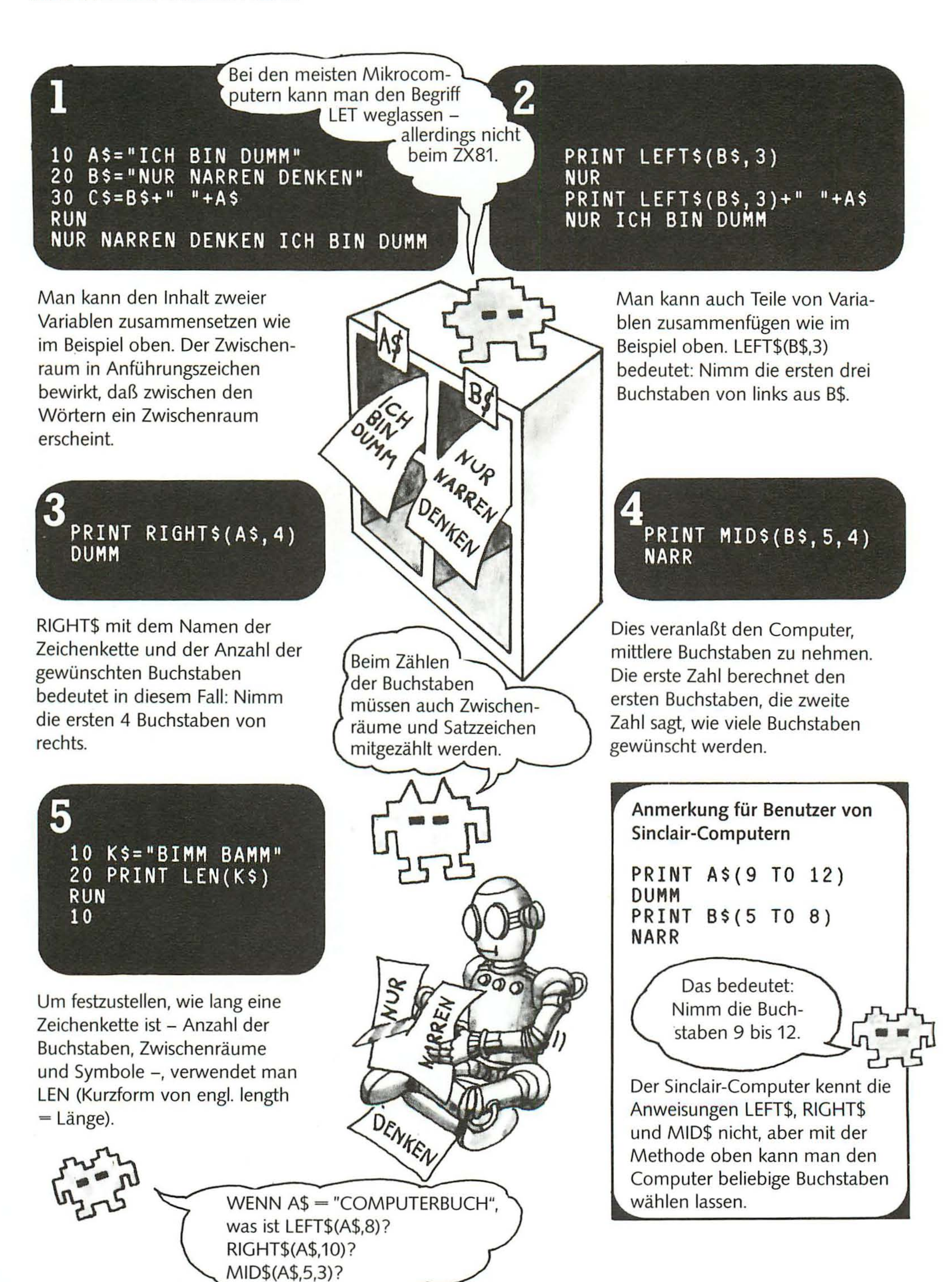
Dies ist das Unterprogramm.

Jedesmal, wenn das Programm wiederholt wird, werden die Wörter und Zahlen in C$ und F durch das nächste Datenpaar ersetzt.

Dieses Programm verwendet ein Unterprogramm zur Überprüfung der Antworten auf die Fragen. Die richtigen Antworten sind in F gespeichert, und die Antworten der befragten Person gehen nach A. In den Zeilen 100 und 110 des Unterprogramms vergleicht der Computer A mit F. Das Wort ABS bedeutet „absoluter Wert"; es bewirkt, daß der Computer die Differenz zwischen den Zahlen A und F feststellt. (Dabei werden Minuszahlen nicht beachtet.) Ist die Differenz kleiner als 10, gibt der Computer OK aus. Ist die Differenz größer als 10, gibt er NEIN aus.

# Spielereien mit Wörtern

Die meisten Computer können die Wörter, die in Variablen gespeichert sind, überprüfen und verschiedene Dinge damit anfangen. So kann der Variablen-Inhalt daraufhin überprüft werden, ob ein bestimmtes Wort oder ein bestimmter Buchstabe enthalten ist. Dies ist dann nützlich, wenn jemand, der das Programm anwendet, Worteingaben überprüfen will. Computer können auch Buchstaben oder Wörter in anderer Reihenfolge zusammensetzen und sie an Buchstaben in anderen Variablen anhängen. Hier wird gezeigt, wie man dies alles in BASIC machen kann.




**1**

```
10 A$="ICH BIN DUMM"
20 B$="NUR NARREN DENKEN"
30 C$=B$+" "+A$
RUN
NUR NARREN DENKEN ICH BIN DUMM
```

Man kann den Inhalt zweier Variablen zusammensetzen wie im Beispiel oben. Der Zwischenraum in Anführungszeichen bewirkt, daß zwischen den Wörtern ein Zwischenraum erscheint.

**2**

```
PRINT LEFT$(B$,3)
NUR
PRINT LEFT$(B$,3)+" "+A$
NUR ICH BIN DUMM
```

Man kann auch Teile von Variablen zusammenfügen wie im Beispiel oben. LEFT$(B$,3) bedeutet: Nimm die ersten drei Buchstaben von links aus B$.

**3**

```
PRINT RIGHT$(A$,4)
DUMM
```

RIGHT$ mit dem Namen der Zeichenkette und der Anzahl der gewünschten Buchstaben bedeutet in diesem Fall: Nimm die ersten 4 Buchstaben von rechts.

**4**

```
PRINT MID$(B$,5,4)
NARR
```

Dies veranlaßt den Computer, mittlere Buchstaben zu nehmen. Die erste Zahl berechnet den ersten Buchstaben, die zweite Zahl sagt, wie viele Buchstaben gewünscht werden.

**5**

```
10 K$="BIMM BAMM"
20 PRINT LEN(K$)
RUN
10
```

Um festzustellen, wie lang eine Zeichenkette ist – Anzahl der Buchstaben, Zwischenräume und Symbole –, verwendet man LEN (Kurzform von engl. length = Länge).

**Anmerkung für Benutzer von Sinclair-Computern**

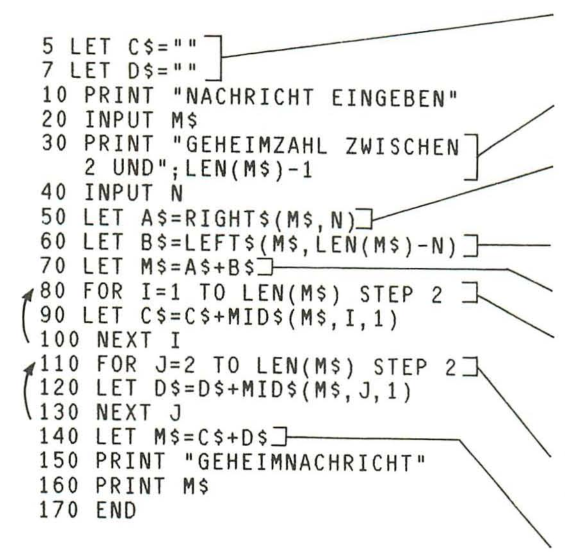
```
PRINT A$(9 TO 12)
DUMM
PRINT B$(5 TO 8)
NARR
```

Das bedeutet: Nimm die Buchstaben 9 bis 12.

Der Sinclair-Computer kennt die Anweisungen LEFT$, RIGHT$ und MID$ nicht, aber mit der Methode oben kann man den Computer beliebige Buchstaben wählen lassen.

## Programm "Geheimcode"

Mit diesem Programm kann man Wörter verschlüsseln. Ähnliche, allerdings viel kompliziertere Programme benutzen auch Geheimdienste, um Codes zu schaffen oder zu knacken.
Am leichtesten versteht man dieses Programm, wenn man eine geheime Nachricht auf ein Blatt Papier schreibt, dann die Programmzeilen durcharbeitet und das tut, was der Computer mit der Nachricht macht; das sollte man aufschreiben.



```
5 LET C$=""
7 LET D$=""
10 PRINT "NACHRICHT EINGEBEN"
20 INPUT M$
30 PRINT "GEHEIMZAHL ZWISCHEN
   2 UND";LEN(M$)-1
40 INPUT N
50 LET A$=RIGHT$(M$,N)
60 LET B$=LEFT$(M$,LEN(M$)-N)
70 LET M$=A$+B$
80 FOR I=1 TO LEN(M$) STEP 2
90 LET C$=C$+MID$(M$,I,1)
100 NEXT I
110 FOR J=2 TO LEN(M$) STEP 2
120 LET D$=D$+MID$(M$,J,1)
130 NEXT J
140 LET M$=C$+D$
150 PRINT "GEHEIMNACHRICHT"
160 PRINT M$
170 END
```

Leere String-Variablen werden bereitgestellt.

Dies bedeutet: Länge der Nachricht minus 1.

N (die geheime Zahl) Buchstaben vom rechten Teil von M$.

Die Länge von M$ minus N Buchstaben vom linken Teil von M$, d. h. der Rest der Buchstaben.

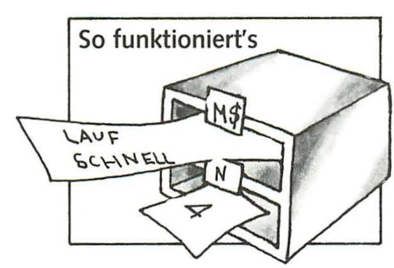
Ersetzt die Buchstaben in M$ durch A$+B$.

Von 1 bis zur Buchstabenanzahl der Nachricht in Zweierschritten, d. h. 1, 3, 5 usw. Bei jeder Wiederholung der I-Schleife wird in Zeile 90 ein Buchstabe aus Position I von M$ nach C$ gebracht.

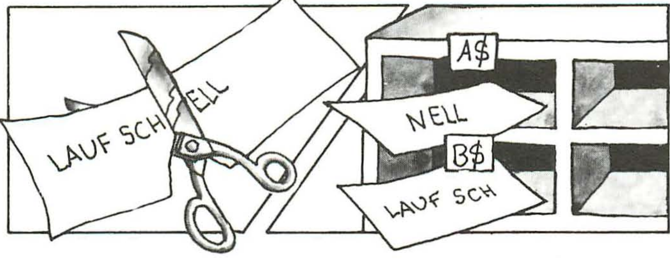
Von 2 bis zur Buchstabenanzahl der Nachricht in Zweier-Schritten, d. h. 2, 4, 6 usw. Funktioniert wie die I-Schleife.

Ersetzt nochmals die Buchstaben in M$.

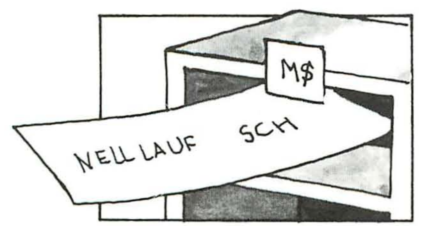
### So funktioniert's



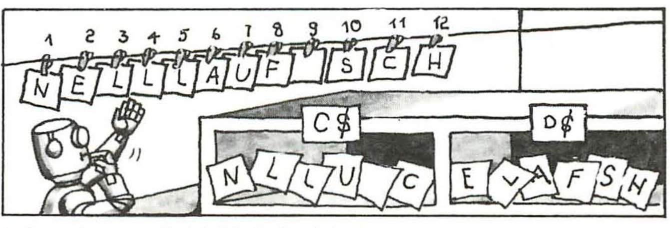

Angenommen, die Nachricht heißt „Lauf schnell", und die Geheimzahl ist 4. Diese Angaben werden in M$ und N gespeichert.




In den Zeilen 50 und 60 benutzt der Computer die Geheimzahl zur Aufteilung der Nachricht. In Zeile 50 werden die vier rechten Buchstaben der Nachricht genommen und in A$ gespeichert. In Zeile 60 wird der Rest der Nachricht in B$ gespeichert.



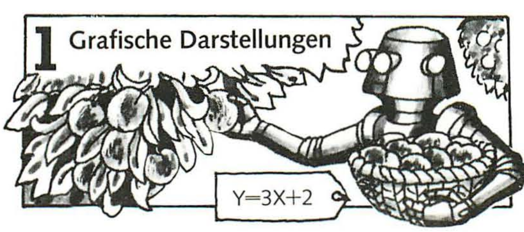

In Zeile 70 werden A$ und B$ zusammengesetzt. Die Buchstaben vom Ende der Nachricht werden nach vorn gesetzt.



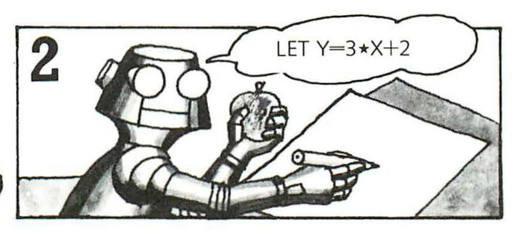

Jedesmal, wenn die I-Schleife durchlaufen wird, wird ein Buchstabe in C$ gespeichert, der sich an ungerader Stelle befindet (also N, L, L usw.). Jedesmal, wenn die J-Schleife durchlaufen wird, wird ein Buchstabe in D$ gespeichert, der sich an gerader Stelle befindet (also E, L, A usw.). Dann werden C$ und D$ zur verschlüsselten Nachricht zusammengesetzt.

# Graphen und Symbole

Man kann einen Computer so programmieren, daß er Informationen auf verschiedene Weise ausgibt, z. B. als Wörter, Zahlen, Bilder oder Grafiken. Durch die Verwendung von Grafiken, Bildern und Symbolen kann man schwierige Informationen leichter verständlich machen.




Man stelle sich einen Pfirsichbaum vor, dessen Fruchternte entsprechend seinem Alter von Jahr zu Jahr wächst. Das kann man mit einer Gleichung ausdrücken, z. B. $Y = 3X + 2$ (Y ist die Ernte, X ist das Alter). Dies kann man sich schwer vorstellen, die grafische Darstellung (ein Graph) ist daher hilfreich.




Mit einem Computer ist es sehr einfach, das Wachsen der Ernte Y in bezug auf das Alter X grafisch darzustellen. Um den Graphen zu zeichnen, muß man den Wert für Y für jeden Wert von X finden. Dies liefert auf einfache Weise die Anweisung LET Y = 3*X+2.




So sieht das Programm für diesen Graphen aus: Die Schleife setzt X auf alle Werte zwischen 1 und 4. Bei jedem Durchgang durch die Schleife verwendet die Zeile 20 den gerade gültigen Wert für X, um Y zu berechnen. Zeile 30 zeichnet X und Y auf den Bildschirm. Bei Programmen für das Zeichnen von Graphen muß man sicherstellen, daß die maximalen Werte für X und Y auf dem Bildschirm liegen, andernfalls gibt es einen Fehler.

### Computer und Mathematik

Bei Rechnungen, die wie 3 x X + 2 aus verschiedenen Teilen bestehen, führt der Computer immer zuerst die Multiplikationen und Divisionen aus, bevor er addiert oder subtrahiert. Der Computer würde also bei den folgenden Aufgaben jeweils übereinstimmende Ergebnisse ausgeben.

```
PRINT 4*6+8          PRINT 8+4*6
32                   32
```

Soll der Computer die Aufgabe in anderer Reihenfolge ausrechnen, dann benutzt man Klammern, z. B. so:

```
PRINT (8+4)*6
72
```

Jetzt addiert der Computer 8 und 4 und multipliziert das Ergebnis mit 6.




Schreiben Sie ein Programm für diesen bekannten Zahlentrick. (Man bestimmt die gedachte Zahl, indem man vom Rechenergebnis 4 abzieht und die dann erhaltene Zahl durch 2 teilt.)

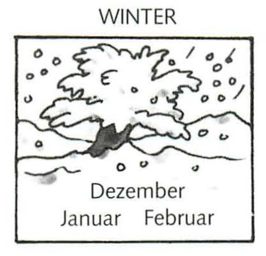
## Geburtstags-Programm

Dieses Programm veranschaulicht eine andere Art der Darstellung von Informationen auf dem Bildschirm. Es benutzt Symbole, um die Anzahl von Menschen, die in den verschiedenen Jahreszeiten geboren sind, zu vergleichen. Ein solches Programm kann z. B. das Auftreten einer bestimmten Vogelart in verschiedenen Jahreszeiten vergleichen oder die Anzahl der gewonnenen Spiele verschiedener Fußballmannschaften. Bei einem Programm dieser Länge empfiehlt es sich, zunächst einen Programmplan anzufertigen.

## Programmplan

**Ziel:** Vergleich der Anzahlen von Menschen, die im Winter, Frühling, Sommer und Herbst geboren wurden.

WINTER



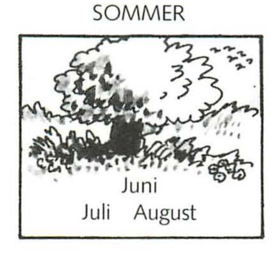
Dezember
Januar Februar

FRÜHLING



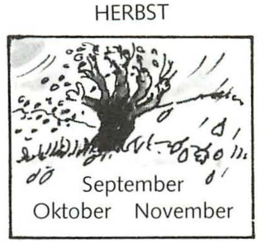
März
April Mai

SOMMER



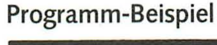
Juni
Juli August

HERBST



September
Oktober November

1. Eingabe der Daten (d. h. der Jahreszeiten, in denen die Menschen geboren wurden).
2. Speicherung der Daten im Computer.
3. Ausgabe der Daten auf dem Bildschirm.



## Programm-Beispiel





## Das Programm

```
5 LET A=0
6 LET B=0
7 LET C=0
8 LET D=0
10 FOR I=1 TO 20
20 PRINT "PERSON ";I;" GEBOREN IM"
30 PRINT "WINTER, FRÜHLING, SOMMER ODER HERBST?"
40 PRINT "DRUCKE W, F, S ODER H"
50 INPUT B$
60 IF B$="W" THEN LET A=A+1
70 IF B$="F" THEN LET B=B+1
80 IF B$="S" THEN LET C=C+1
90 IF B$="H" THEN LET D=D+1
100 NEXT I
110 PRINT "IM WINTER  : ";
115 LET N=A
120 GOSUB 200
130 PRINT "IM FRÜHLING: ";
135 LET N=B
140 GOSUB 200
150 PRINT "IM SOMMER  : ";
155 LET N=C
160 GOSUB 200
170 PRINT "IM HERBST  : ";
175 LET N=D
180 GOSUB 200
190 STOP
200 REM UNTERPROGRAMM STERNE
210 IF N=0 THEN GOTO 250
220 FOR I=1 TO N
230 PRINT "*";
240 NEXT I
250 PRINT
260 RETURN
```

Zeilen 5–8: Leere Variablen für die Gesamtzahl für jede Jahreszeit.

Zeile 10: Schleife zur Befragung jeder einzelnen Person.

Die Zeilen 60 bis 90 prüfen die Antwort in B$ und addieren 1 zur Variablen der betreffenden Jahreszeit.

Zeile 100: Schickt den Computer zurück zur weiteren Fragestellung.

Zeile 120: Das Unterprogramm bewirkt, daß der Computer so viele Sterne ausdruckt, wie die Anzahl in jeder Variablen ist.

Zeilen 135–155: Durch die Speicherung der jeweiligen Gesamtzahl in N kann dasselbe Unterprogramm für jede Jahreszeit verwendet werden.

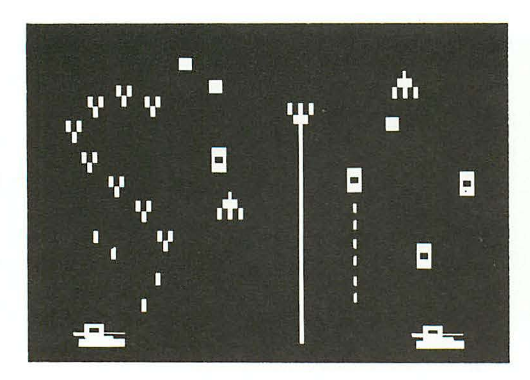
Zeile 170: Bewirkt das Ausdrucken der Sterne in derselben Zeile.

Zeile 210 ist für den Fall da, daß niemand in einer bestimmten Jahreszeit geboren wurde.

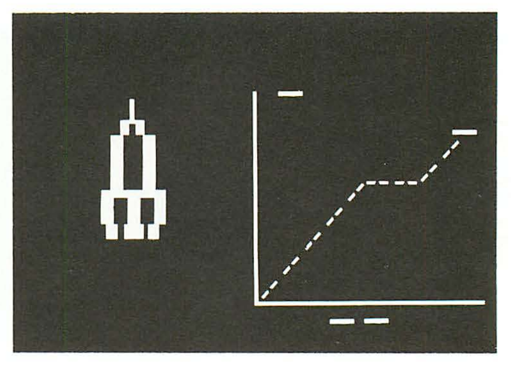
Zeile 220: Im Hauptprogramm wird N gleich der Gesamtzahl für A, B, C oder D gesetzt. Die Schleife bewirkt, daß der Computer die Zeile 230 „N"-mal ausführt.

# Bewegte Bilder

Auf diesen beiden Seiten wird gezeigt, wie man PLOT und UNPLOT anwenden kann, um auf dem Bildschirm bewegte Bilder zu erzeugen. Bewegte Bilder nennt man *Animationsgrafik*. Sie sind nützlich für Spiele oder zur Illustration von Programmen, die z. B. das Schwerkraftprinzip oder Flugbahnen erläutern.



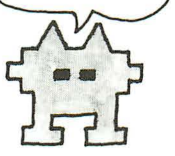
Bei Video- und Computerspielen wird die Grafik von einem kleinen Computer gesteuert. Der Computer ist so programmiert, daß er nur diese Spiele beherrscht. Die Programme sind nicht in BASIC geschrieben, sondern im Code des Computers.



Ein Mikrocomputer, der in BASIC „für den Hausgebrauch" programmiert ist, kann nur einfachere und langsamere Bilder erzeugen. Er kann die Befehle an den Bildschirm nicht rasch genug bewältigen, um wirklich schnell bewegte Grafik zu zeichnen.

## 1 Zeichen-/Lösch-Programm

```
10 LET X=1
20 LET Y=1
30 PLOT(X,Y)
40 UNPLOT(X,Y)
50 LET X=X+1
60 LET Y+Y+1
70 GOTO 30
```



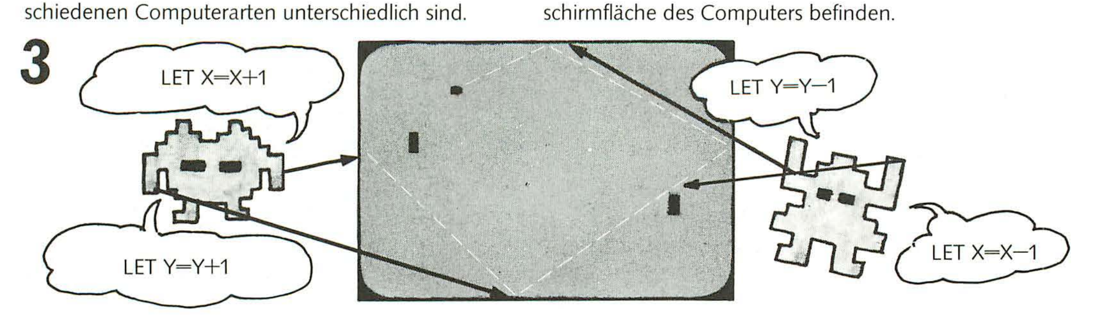

Dieses kurze Programm läßt einen Lichtpunkt über den Bildschirm wandern. Denken Sie daran, daß die Befehle für PLOT und UNPLOT bei den verschiedenen Computerarten unterschiedlich sind.

## 2



Sobald der Punkt eine Ecke des Bildschirms erreicht, stoppt das Programm mit einer Fehlermeldung, da sich die Werte von X und Y außerhalb der Bildschirmfläche des Computers befinden.

## 3

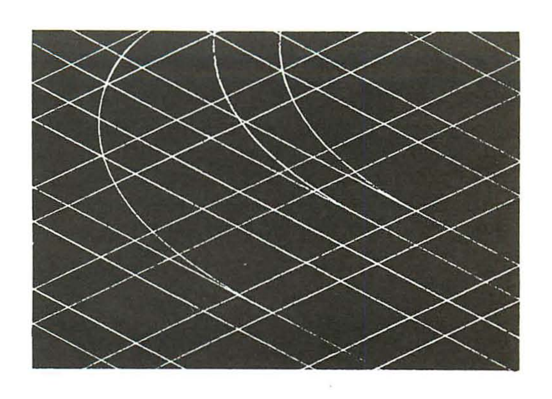



Video-Feldspiele verwenden Programme wie das oben gezeigte, um den Ball auf dem Bildschirm zu bewegen. Es gibt einfache Programmregeln, die den Ball in Bewegung halten, sobald er den Bildschirmrand erreicht hat.

Berührt der Ball den oberen Bildrand, dann wird der zu Y zu addierende Betrag statt dessen abgezogen. Analog wird bei Berührung des rechten Bildrandes der Betrag von X abgezogen.

## Programm für Linienmuster

Dieses Programm zeichnet eine Linie über den Bildschirm. Wenn die Linie einen Rand erreicht, wird sie in einer anderen Richtung weitergeführt. UNPLOT wird nicht verwendet, daher hinterlassen die Linien ein Muster auf dem Bildschirm. Das Bild rechts zeigt das Ergebnis nach dem Programmstart. Zeile 100 des Programms bewirkt die Zeichnung von 10 000 Bildpunkten. Man kann diese Anzahl verändern, um den Vorgang zu verkürzen; man kann auch BREAK benutzen, um das Programm abzubrechen, wenn ein gewünschtes Muster erreicht ist.



```
10 REM FALLS NOTWENDIG HIER ANWEISUNG FÜR GRAFIK-MODUS
20 PRINT "ZAHL DER BILDPUNKTE BREITE?";
30 INPUT B
40 PRINT "ZAHL DER BILDPUNKTE HÖHE?";
50 INPUT H
55 CLS
```

Die Zeilen 20 bis 50 fragen nach Höhe und Breite des Bildschirms. Das Semikolon stellt die Antwort in dieselbe Zeile wie die Frage.

```
60 LET X=B/2
70 LET Y=H/2
```

Dies läßt X und Y im Bildschirm-Mittelpunkt beginnen.

```
80 LET S=1
90 LET T=1
```

S und T sind die Größen, die zu X und Y addiert werden, damit die Linie sich bewegt.

```
100 FOR I=1 TO 10000
```

Die Schleife von Zeile 100 bis 190 wird 10 000mal wiederholt. Jedesmal werden X und Y um einen winzigen Betrag verändert.

```
110 LET S=S+(INT(RND(1)*10+1)-5)/50
120 LET X=X+S
130 LET Y=Y+T
```

Dies bewirkt, daß bei jedem Schleifendurchlauf ein Betrag zu X addiert wird, der um einen kleinen Betrag schwankt.

```
140 IF X<5 THEN LET S=-S
150 IF X>B-5 THEN LET S=-S
160 IF Y<5 THEN LET T=-T
170 IF Y>H-5 THEN LET T=-T
```

Diese Zeilen sind der Randtest. S und T ändern die Richtung, sobald X und Y sich bis auf 5 Bildpunkte einem Rand nähern.

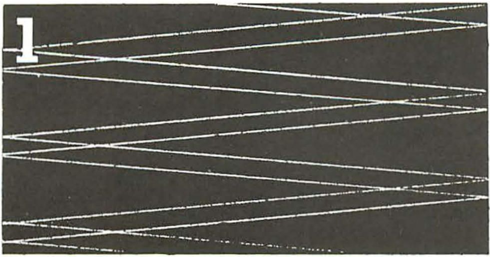
```
180 GOSUB 300
190 NEXT I
200 STOP
```

Hier wird der Computer zum Unterprogramm des Linienzeichnens geschickt.

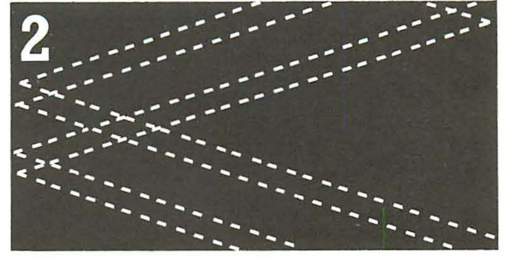
```
300 REM PLOT LINE
310 PLOT(X,Y)
320 RETURN
```

Zeichnet den Bildpunkt mit dem gerade gültigen Wert für X und Y.

### Experimente




Zeile 110 addiert zu X jedesmal eine winzige Zufallszahl. Daher „wackelt" die Linie über den Bildschirm. Wenn Sie einen Computer haben, versuchen Sie, diese Linie zu löschen. Die Linien auf dem Bildschirm sollten parallel laufen.




Versuchen Sie, die Zahlen in Zeile 80 und 90 zu verändern, z. B. 5 oder 10 (oder noch größer bei einem Computer mit hochauflösender Grafik). Dadurch entstehen unterbrochene Linien.

# Verse aus dem Computer

Auf den folgenden Seiten wird ein Programm vorgestellt, mit dem man eine Reihe von „*Gedichten*“ erzeugen kann. Eine Fassung dieses Programms ist im „*Ersten Buch der Elektronik*“ (Otto Maier Verlag, 1983) erschienen. Dort wird der Bau eines „Papier-Computers“ gezeigt. Das gleiche Programm wird hier in einer BASIC-Version vorgestellt.

**Programm für den Papier-Computer**

Papier-Computer

6. DREHE DEN KREISEL, UM N ZU FINDEN.

Zahlenkreisel

```
1   A=0 und B=0
2   Erhöhe A um 1
3   Wenn A=6 gehe zu Zeile 10
4   Schreibe Datenzeile A
5   Erhöhe B um 1
6   Drehe Kreisel und ermittle N
7   Schreibe Datenwörter aus Zeile B,
    Spalte N
8   Wenn B=3 oder 5, gehe zu Zeile 5
9   Gehe zu Zeile 2
10  Stop
```

**Datenzeilen**

```
ES WAR EINMAL EIN MANN AUS
DER
SEIN
DOCH DANN IN DER NACHT
ER FAND NIEMALS HERAUS
```

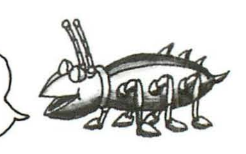
**Datenwörter**

| | | | | |
|---|---|---|---|---|
| 1 | Groß Rist | Bad Nist | Neu Wist | Klein Quist |
| 2 | versteckte | bemalte | zeigte | bedeckte |
| 3 | Bein | Hemd | Haus | Schaf |
| 4 | in Batist | als Tourist | Das war Mist | und tanzt Twist |
| 5 | flog es weg | wurd' es blau | platzte es | fiel es um |
| 6 | daß es kracht | mit Bedacht | und es lacht | grad' um acht |
| 7 | wo es ist | was es frißt | was es ist | was ihr wißt |

```
Es war einmal ein Mann aus Neu Wist
der versteckte sein Schaf als Tourist
doch dann in der Nacht
wurd' es blau und es lacht
er fand niemals heraus
```

Dies ist das Programm für den Papier-Computer. Es sieht so aus, als sei es in BASIC geschrieben, aber es läuft nicht auf einem echten Computer. Wörter und Sätze sind auf Papierstückchen geschrieben; das Programm weist an, welche davon auszuwählen sind. Der Zahlenkreisel liefert Zufallszahlen zwischen 1 und 4.

## 1 Übertragung des Programms in BASIC



So läuft es noch nicht auf dem Computer.

```
10 LET A=0
20 LET B=0
30 LET A=A+1
40 IF A=6 THEN STOP
50 Schreibe Datenzeile A
60 LET B=B+1
70 LET N=INT(RND(1)*4+1)
80 Schreibe Datenwörter aus Zeile B,
   Spalte N
90 IF B=3 THEN GOTO 60
100 IF B=5 THEN GOTO 60
110 GOTO 30
120 END
```

- Zeilen 10–20: Diese Zeilen setzen die Variablen gleich Null.
- Zeilen 30–40: Die Zeilen 30 und 40 zählen die vom Computer ausgewählten Datenzeilen.
- Zeile 50: Die Zeilen 50 und 80 sind noch nicht in BASIC.
- Zeile 60: Zeile 60 zählt die Datenwörter.
- Zeile 70: Erzeugt eine Zufallszahl zwischen 1 und 4.
- Zeilen 90–100: Die Zeilen 90 und 100 setzen zurück, um eine andere Datenzeile auszuwählen.

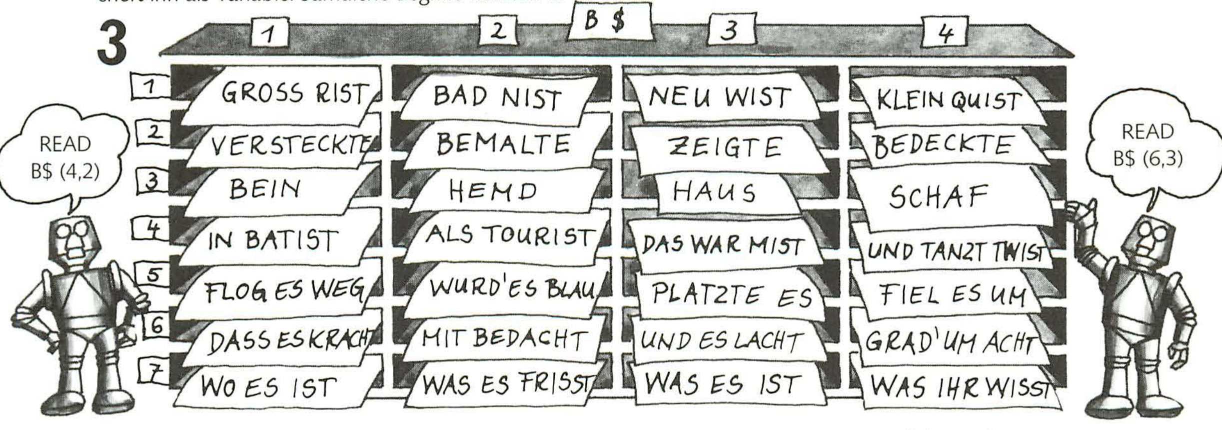
Der größte Teil des Programms läßt sich leicht in BASIC übertragen; schwieriger ist das mit den Zeilen 50 und 80. Der Computer muß einen Weg finden, um aus den Datenzeilen die entsprechenden Wörter für jede Zeile des „Gedichts“ auszuwählen.

## 2 Der Computer wird gefüttert

```
50 READ A$
.
.
180 DATA ES WAR EINMAL EIN MANN AUS,
    DER,SEIN
190 DATA DOCH DANN IN DER NACHT,
    ER FAND NIEMALS HERAUS
```

Um den Computer mit den Datenzeilen und Datenwörtern zu versorgen, wird die Anweisung READ... DATA verwendet. Jedesmal wenn der Computer die READ-Anweisung ausführt, nimmt er einen neuen Begriff aus der Datenzeile und speichert ihn als Variable. Sämtliche Begriffe können in einer einzigen Variablen A$ gespeichert werden. Eine Variable, die mehr als einen Begriff enthält, bezeichnet man als *Feld.* Jeder Begriff wird durch eine Nummer identifiziert. So ergibt z. B. READ A$(3) den Begriff „sein"*.

## 3




Eine Variable kann mehrere Datenreihen umfassen. Alle Daten können in einer Variablen abgelegt werden, wie das Bild zeigt. Man spricht dann von einem *zweidimensionalen Feld.* Jeder Begriff wird durch die Reihennummer und die Spaltennummer identifiziert. So ergibt READ B$(4, 2) „als Tourist" und READ B$(6, 3) „und es lacht". Zahlen können als Variablen ebenfalls in Feldern abgelegt werden.

## 4 Daten in Variablen umwandeln




```
10 FOR I=1 TO 7
20 FOR J=1 TO 4
30 READ B$(I,J)
40 NEXT J
50 NEXT I
60 DATA GROSS RIST,BAD NIST,NEU WIST,KLEIN QUIST
70 DATA VERSTECKTE, BEMALTE,ZEIGTE,BEDECKTE
80 DATA BEIN,HEMD,HAUS,SCHAF
90 DATA IN BATIST,ALS TOURIST,DAS WAR MIST,UND TANZT TWIST
100 DATA FLOG ES WEG,WURD' ES BLAU, PLATZTE ES,FIEL ES UM
110 DATA DASS ES KRACHT,MIT BEDACHT,UND ES LACHT,GRAD' UM ACHT
120 DATA WO ES IST,WAS ES FRISST,WAS ES IST,WAS IHR WISST
```

Damit die Daten eingelesen werden können, müssen die Zahlen in Klammern nach dem READ-Befehl geändert werden. Dies kann mittels einer verschachtelten Schleife geschehen. B$ verwendet im Beispiel oben eine I-Schleife für die Reihennummer und eine J-Schleife für die Spaltennummer. Jedesmal wenn die I-Schleife ausgeführt wird, wird die J-Schleife viermal wiederholt – einmal für jede Spalte einer Reihe.



* Sinclair-Computer behandeln Variablen auf eine andere Art, daher läuft dieses Programm nicht auf einem Sinclair. Mehr darüber auf der nächsten Seite.

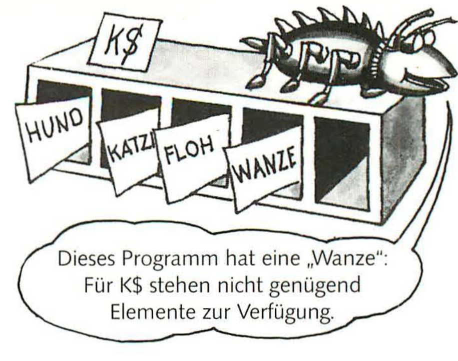
# 5 Speicherplatz für Variablen

```
5 DIM K$(5)
10 FOR I=1 TO 5
20 READ K$(I)
30 NEXT I
40 STOP

60 DATA HUND,KATZE,FLOH,WANZE
```

Dimensionierung der Variablen: 5 Begriffe in einer Reihe. (zu Zeile 5)

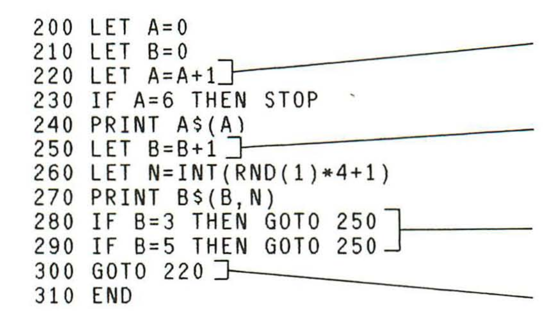
Mit dieser Anweisung werden die Daten in K$ gespeichert, sooft die Schleife wiederholt wird. (zu Zeile 20)




Am Anfang des Programms muß man dem Computer sagen, wieviel Speicherraum er für die Variablen zu reservieren hat. Der DIM-Befehl, gefolgt vom Namen der Variablen und der Zahl der Elemente, z. B. DIM K$(5), *dimensioniert* den Speicherplatz.

Für ein zweidimensionales Feld werden die Anzahl der Reihen und der Spalten angegeben, z. B. DIM C$(5, 3). Die Zahlen müssen der Anzahl der Elemente entsprechen, sonst wird ein Fehler gemeldet.

# 6 Daten ausdrucken



```
200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1
260 LET N=INT(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

A zählt, wie oft dieser Teil des Programms wiederholt wird. (zu Zeile 220)

B zählt die Reihen mit den Wörtern und stellt sicher, daß immer die entsprechende Reihe verwendet wird. (zu Zeile 250)

Die Zeilen 280 und 290 bewirken, daß Wörter aus einer anderen Datenreihe angezeigt werden, bevor die nächste Datenzeile ausgedruckt wird.

Diese Zeile veranlaßt den Computer zurückzugehen, um die nächste Datenzeile anzuzeigen. (zu Zeile 300)

Der Computer benötigt diese Anweisungen, um die Datenzeilen und Datenwörter in der richtigen Reihenfolge anzuzeigen. Dieser Programmteil wird fünfmal wiederholt. Dabei werden jedesmal die Daten der Zeile A und einige Wörter der Reihe B angezeigt. Durch die Zufallszahl N wird festgelegt, welche Wörter ausgewählt werden.

## Sinclair-Computer und Variablen

Das Programm läuft in dieser Form nicht auf Sinclair-Computern, da dort Zeichenketten auf eine andere Art verarbeitet werden.

Bei zweidimensionalen Feldern müssen sowohl die Nummer der Reihe als auch die Nummern der Buchstaben angegeben werden; z. B. A$(2, 5 TO 9) entspricht „Wanze“.







Damit ein Sinclair-Computer weiß, welche Datenelemente er aus dem Datenfeld auswählen soll, muß man ihm die Nummern des ersten und letzten Buchstabens des gewünschten Begriffs angeben. Mit dieser Methode werden bei Sinclair-Computern auch die Anweisungen LEFT$, RIGHT$ u. a. durchgeführt (siehe Seite 32).

Am Anfang des Programms wird angegeben, wieviel Reihen das Feld hat und wieviel Buchstaben in jeder Reihe sind. DIM A$(2, 9) bedeutet: 2 Reihen mit jeweils 9 Buchstaben. Alle Reihen des Feldes müssen die gleiche Anzahl Buchstaben haben.

## Das vollständige Programm

Jetzt können alle Teile des Programms zusammengefaßt und eingetippt werden. Im ersten Abschnitt des Programms (Zeilen 10 bis 190) werden die Daten eingelesen, im zweiten Abschnitt (Zeilen 200 bis 310) wird das Gedicht angezeigt. Jedesmal wenn das Programm gestartet wird, ergibt sich wegen der Zufallszahl N ein neues Gedicht, da der Computer sich andere Wörter aussucht.

```
10 DIM A$(5)
20 DIM B$(7,4)
30 FOR I=1 TO 7
40 FOR J=1 TO 4
50 READ B$(I,J)
60 NEXT J
70 NEXT I
80 DATA GROSS RIST,BAD NIST,NEU WIST,KLEIN QUIST
90 DATA VERSTECKTE, BEMALTE,ZEIGTE,BEDECKTE
100 DATA BEIN,HEMD,HAUS,SCHAF
110 DATA IN BATIST,ALS TOURIST,DAS WAR MIST,UND TANZT TWIST
120 DATA FLOG ES WEG,WURD' ES BLAU, PLATZTE ES,FIEL ES UM
130 DATA DASS ES KRACHT,MIT BEDACHT,UND ES LACHT,GRAD' UM ACHT
140 DATA WO ES IST,WAS ES FRISST,WAS ES IST,WAS IHR WISST
150 FOR I=1 TO 5
160 READ A$(I)
170 NEXT I
180 DATA ES WAR EINMAL EIN MANN AUS,DER,SEIN
190 DATA DOCH DANN IN DER NACHT,ER FAND NIEMALS HERAUS

200 LET A=0
210 LET B=0
220 LET A=A+1
230 IF A=6 THEN STOP
240 PRINT A$(A)
250 LET B=B+1
260 LET N=INT(RND(1)*4+1)
270 PRINT B$(B,N)
280 IF B=3 THEN GOTO 250
290 IF B=5 THEN GOTO 250
300 GOTO 220
310 END
```

Die Zeilen 10 und 20 weisen den Speicherplatz für die Variablen an: 5 Buchstaben für A$ und 7 Reihen zu 4 Buchstaben für B$.

Das sind die verschachtelten Schleifen zum Einlesen der Daten in B$.

In den Zeilen 80 bis 140 stehen alle Wörter, die im Feld B$ zu speichern sind.

Schleife zum Einlesen der Daten in A$.

In den Zeilen 180 und 190 stehen alle Wörter, die im Feld A$ zu speichern sind.

Zeigt die Datenzeile an, die in A$ unter A gespeichert ist.

Zeigt die Wörter an, die in B$ Reihe B, Spalte N gespeichert sind.

Das Programm endet mit Zeile 230, wenn A=6; es kommt also nie zu Zeile 310. Manche Computer benötigen aber als Abschluß den Befehl END.

### Beispiele

```
ES WAR EINMAL EIN MANN AUS
KLEIN QUIST
DER
BEDECKTE
SEIN
HEMD
DAS WAR MIST
DOCH DANN IN DER NACHT
FLOG ES WEG
MIT BEDACHT
ER FAND NIEMALS HERAUS
WAS IHR WISST
```

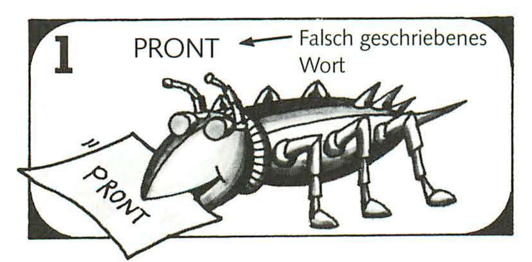
```
ES WAR EINMAL EIN MANN AUS
BAD NIST
DER
BEMALTE
SEIN
HAUS
IN BATIST
DOCH DANN IN DER NACHT
PLATZTE ES
DASS ES KRACHT
ER FAND NIEMALS HERAUS
WO ES IST
```

Hier sind zwei Versionen der 16 384 möglichen Gedichte. Wenn immer wieder die gleichen Gedichte angezeigt werden, sollte man im Handbuch nachlesen, wie der Computer unterschiedliche Zufallszahlen erzeugt. Manche Computer erzeugen nämlich nach dem Einschalten jedesmal die gleiche Folge von Zufallszahlen.

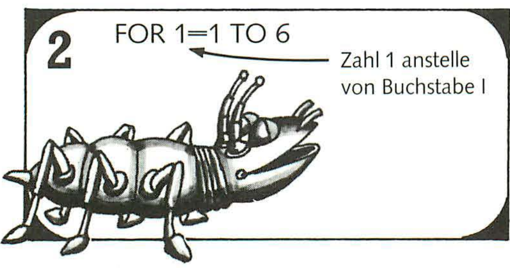
# Hinweise zum Programmieren

Auf diesen beiden Seiten sind einige Hinweise zusammengestellt, die das eigene Programmieren erleichtern. Außerdem ist eine Liste der häufigsten Fehler und ihrer Ursachen aufgeführt. Die wahrscheinlichsten Fehler kommen zuerst. Sollte ein Programm also nicht laufen, dann empfiehlt es sich, das Programm auf mögliche Fehler zu überprüfen.

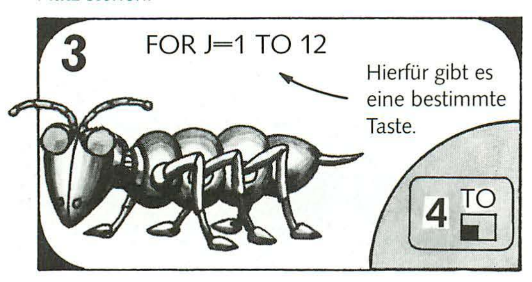
## Fehlersuche




Zunächst sollten BASIC-Ausdrücke auf Tippfehler untersucht werden. BASIC-Ausdrücke mit Tippfehlern kann der Computer nicht erkennen.




Überprüfen Sie alle O (Buchstabe) und 0 (Ziffer Null) sowie I (Buchstabe) und 1 (Ziffer), um sicherzustellen, daß die richtigen Zeichen am richtigen Platz stehen.




Beim Sinclair-Computer müssen die vorhandenen Tasten für die BASIC-Wörter benutzt werden; die Wörter werden also nicht Buchstabe für Buchstabe eingetippt.

## Programme schreiben

Beim Schreiben von Programmen sollte man immer an die drei Haupttätigkeiten eines Computers denken: Ein Computer kann einfache Anweisungen ausführen, Vorgänge wiederholen und Entscheidungen treffen. Diese sind die Bausteine aller Programme.




```
LET A=3
LET N=N+1
PRINT A/T
PLOT(X,Y)
```



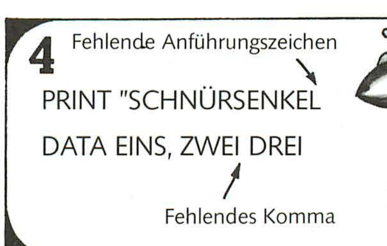

```
FOR J=1 TO 6

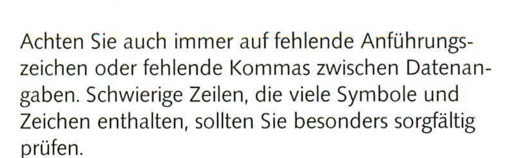
20 LET A=1
30 IF A<10 THEN
   GOTO 100
```




```
IF X=Y THEN STOP
IF K$="HALLO"
   THEN PRINT A
```

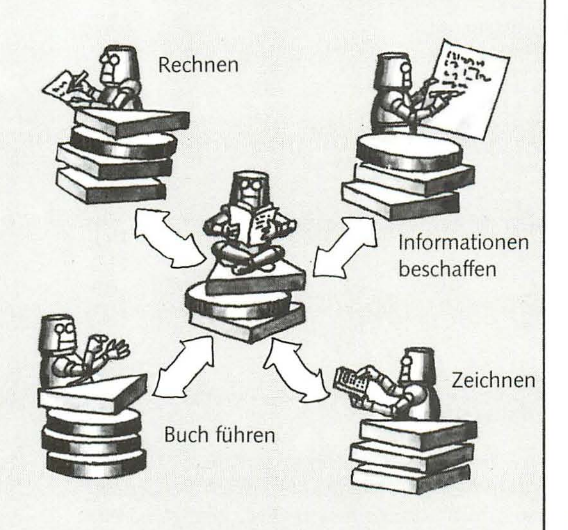
In diesem Buch sind die wesentlichsten BASIC-Befehle behandelt, die genügen, um den Computer diese Tätigkeiten ausführen zu lassen. Beim Schreiben von Programmen sollte man überlegen, was der Computer an einem bestimmten Punkt tun muß; danach richtet sich dann die Entscheidung über die richtigen Befehle.



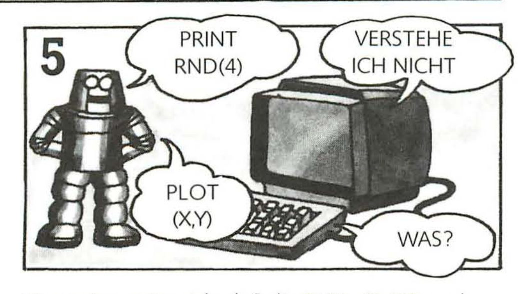



Achten Sie auch immer auf fehlende Anführungszeichen oder fehlende Kommas zwischen Datenangaben. Schwierige Zeilen, die viele Symbole und Zeichen enthalten, sollten Sie besonders sorgfältig prüfen.

Normalerweise gibt es mehrere Möglichkeiten, ein Programm zu schreiben; manche sind vielleicht kürzer und „eleganter" als andere. Schreibt man ein langes Programm, empfiehlt es sich, das Programm zu unterteilen und Unterprogramme für jede Aufgabe einzubauen. Das Kernprogramm kann aus einfachen Anweisungen, Entscheidungen und Wiederholungen bestehen und bestimmen, wann und wie oft der Computer die Unterprogramme ausführen soll.



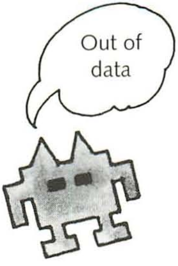

Eine solche Aufteilung von Programmen in einzelne Bereiche erleichtert die Fehlersuche. Dabei kann jeder Bereich getrennt überprüft werden, ohne daß man das ganze Programm ablaufen lassen muß. Man sollte auch daran denken, jeden Bereich mit einer REM-Zeile zu kennzeichnen, die seinen Zweck angibt.




Vergewissern Sie sich, daß die RND-, PLOT- und CLS-Befehle richtig eingegeben werden. Überprüfen Sie ferner, ob Sie dem Computer eine allgemeine Grafik-Zeile eingegeben haben, falls er eine braucht.

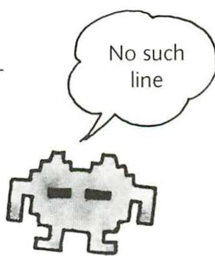
## Fehlermeldungen

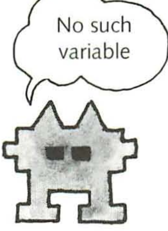
Alle Computer geben bei fehlerhaften Programmen Fehlermeldungen. Diese Meldungen sind im Computer-Handbuch erklärt. Hier sind einige der häufigsten Fehlermeldungen zusammengestellt.




◀ Dies bedeutet, daß in der DATA-Zeile nicht ausreichend viele Daten für den Computer zu lesen sind. Die Ursache dafür kann z. B. ein fehlendes Komma zwischen zwei Daten sein, so daß der Computer diese zwei Daten als ein Datum auffaßt.

▶ Eine Zeile mit dieser in einer GOTO- oder GOSUB-Anweisung angegebenen Nummer existiert nicht. Die Ursache dafür kann ein Tippfehler sein, oder Sie haben vielleicht die Zeile gelöscht, indem Sie eine neue Zeile mit der gleichen Nummer eingegeben haben.

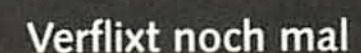



◀ Diese Anzeige kann z. B. bei einem Sinclair-Computer erscheinen. Sie bedeutet, daß eine bestimmte Variable nicht in einer Zeile wie in LET C=0 oder LET C="" definiert wurde.

▶ Dies bedeutet, daß in einer Schleife die NEXT-Zeile fehlt. Ursache kann ein falsch eingegebener Variablen-Name sein, den der Computer nicht erkennt, oder ein Tippfehler, z. B. I statt 1.

FOR without NEXT



**Verflixt noch mal**

Manche Fehler sind sehr schwer zu finden. Wenn das Programm jedoch nicht läuft, muß irgendwo ein Fehler stecken. Ist der Fehler unauffindbar, empfehlen wir, schwierige Zeilen nochmals einzugeben. Beim zweiten Mal werden sie dann vielleicht doch richtig, ohne daß man merkt, wo der Fehler nun eigentlich gesteckt hat.

# Lösungen

### Seite 15

## Programm "Name und Nachricht"

```
10 PRINT "WIE HEISSEN SIE?"
20 INPUT N$
30 PRINT "HALLO"
40 PRINT N$
50 PRINT "WIE GEHT ES IHNEN?"
```

### Seite 17

## 1. Rechenprogramm

```
10 LET A=9
20 LET B=7
30 PRINT A*B
40 PRINT A/B
50 LET A=A+1
60 LET B=B+3
70 PRINT A*B,A/B
80 END
```

Komma für Zwischenraum

Leerstellen

## 2. Einmaleins-Programm

```
30 PRINT A;" MAL ";B;" IST ";A*B
40 PRINT A;" GETEILT DURCH ";B;"IST ";A/B
```

## 3. Programm "Name und Nachricht"

(Abwandlung)

```
10 PRINT "WIE HEISSEN SIE?"
20 INPUT N$
30 PRINT " HALLO ";N$;" WIE GEHT ES IHNEN?"
```

### Seite 18

## Rechenprogramm

```
10 PRINT "WIEVIEL IST 7 MAL 7?"
20 INPUT A
30 IF A=49 THEN PRINT "STIMMT!"
40 IF A<>49 THEN PRINT "NEIN!";7*7
```

Das Semikolon an dieser Stelle ist wichtig!

### Seite 19

## Alter-Rate-Programm

Man ersetze Zeile 30 und füge eine neue Zeile 35 hinzu:

```
30 IF G<14 THEN PRINT "ÄLTER ALS"
35 IF G>14 THEN PRINT "JÜNGER ALS"
```

### Seite 23

## Grafik-Programm

```
5 LET C=0
45 LET C=C+1
50 IF C<6 THEN GOTO 10
```

## Gezeichnete Initialen

Beispielprogramm für das Zeichnen des Buchstabens L.

```
10 LET X=15
20 LET Y=30
30 PLOT (X,Y)
40 LET Y=Y-1
50 IF Y>5 THEN GOTO 30
60 LET X=X+1
70 PLOT (X,Y)
80 IF X<45 THEN GOTO 60
90 END
```

### Seite 24

## Zufallszahlen

Die Formel für eine Zufallszahl zwischen 10 und 20 ist INT(RND(1)*11+9). Bei Computern, bei denen eine Zahl in Klammern hinter RND genügt, wäre es RND(11)+9. Da es elf Zahlen zwischen 10 und 20 gibt, muß man Zufallszahlen von 1 bis 11 wählen und 9 hinzuzählen.

### Seite 25

## Angriff im Weltraum

Hier sind die Zeilen, die man hinzufügen muß, um die Trefferzahl zu zählen.

```
15 LET S=0
75 IF X=A*B THEN LET S=S+1
95 PRINT "GETROFFEN ";S;" VON 6 GEGNERN"
```

### Seite 27

## 1. Achterreihen-Programm

```
10 PRINT "ACHTERREIHE"
20 FOR J=1 TO 12
30 PRINT J;"* 8 =";J*8
40 NEXT J
```

Seite 27

## 2. 1malN-Reihe

```
10 INPUT "ZAHL EINGEBEN: ";N
20 PRINT "ES FOLGT DIE ";N;"-REIHE"
30 FOR I=1 TO 12
40 PRINT I;" MAL ";N;" IST ";I*N
50 NEXT I
60 INPUT "NEUE ZAHL EINGEBEN (J/N)";M$
70 IF M$="J" THEN GOTO 10
```

Beim ZX81 benötigt man getrennte PRINT- und INPUT-Zeilen.

Seite 34

## Zahlentrick-Programm

```
10 PRINT "MAN DENKE SICH EINE ZAHL,"
20 PRINT "VERDOPPELE SIE, ADDIERE 4,"
30 PRINT "DIVIDIERE DURCH 2, ADDIERE 7,"
40 PRINT "MULTIPLIZIERE MIT 8, SUBTRAHIERE 12,"
50 PRINT "DIVIDIERE DURCH 4 UND SUBTRAHIERE 11."
60 PRINT "WIE HEISST DAS ERGEBNIS?"
70 INPUT N
80 PRINT "DIE GEDACHTE ZAHL IST";(N-4)/2
```



Die Klammern sind für die richtige Rechenreihenfolge wichtig.

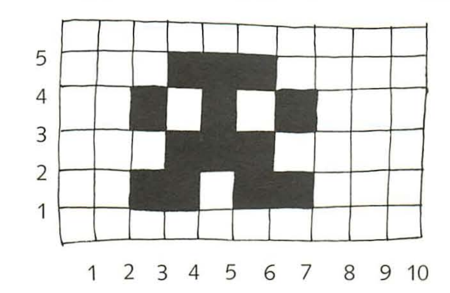
Seite 32

## Computerbuch-Worträtsel

```
LEFT$(A$,8) ist "COMPUTER"
RIGHT$(A$,10) ist "PUTER BUCH"
MID$(A$,5,8) ist "UTER BUC"
```

## Raumfahrer-Programm

**1**




Hier ist ein kleiner „Raumfahrer" auf kariertes Papier gezeichnet.

**2**

4,5; 5,5; 6,5

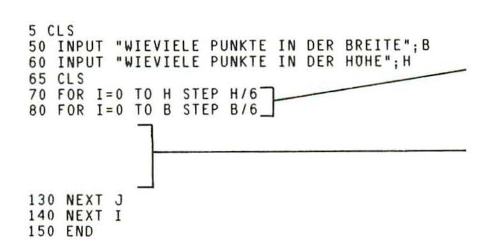
3,4; 5,4; 7,4

4,3; 5,3; 6,3

3,2; 4,2; 6,2; 7,2

Dies sind die Koordinaten derjenigen Quadrate, aus denen die Figur besteht.

**3**



```
5 CLS
50 INPUT "WIEVIELE PUNKTE IN DER BREITE";B
60 INPUT "WIEVIELE PUNKTE IN DER HÖHE";H
65 CLS
70 FOR I=0 TO H STEP H/6
80 FOR I=0 TO B STEP B/6

130 NEXT J
140 NEXT I
150 END
```

Man kann die 6 auch in eine größere Zahl abändern, dann erscheinen entsprechend mehr Raumfahrerfiguren auf dem Bildschirm. (Ist die Zahl jedoch zu groß, gibt es einen Fehler.)

Die Zeichen-Zeilen gehören hierhin, z. B.
90 PLOT (J+3, I+2)
92 PLOT (J+4, I+2),
für den Fall der beiden Quadrate unten links in der Figur oben. Für jedes Quadrat ist eine Programmzeile erforderlich.

Schreiben Sie das Muster-Wiederholungsprogramm ab, und zwar ohne die Zeilen 10 bis 40 und 90 bis 140, wie oben gezeigt. (Diese Zeilen stellen das Zufallsmuster des Programms her, werden also nicht benötigt.)
Fügen Sie nun Ihre eigenen Zeichen-Zeilen in das Programm ein, und zwar zwischen die Zeilen 80 und 140. (Sie können die Programmzeilen auch neu numerieren.)
Um den „Raumfahrer" zu wiederholen, muß man bei jedem Koordinaten-Paar zur ersten Zahl J und zur zweiten Zahl I hinzuzählen.

# Die wichtigsten BASIC-Ausdrücke

Hier ist eine Liste der BASIC-Ausdrücke, die in diesem Buch verwendet werden. Jeder Ausdruck wird kurz erklärt. Einige Ausdrücke, wie CLS, gehören nicht zur Standardausrüstung von Computern, sie sind mit einem Sternchen gekennzeichnet. Diese Befehle sollten im Computer-Handbuch überprüft werden, wenn man einen Computer hat.

★ BREAK (abbrechen): Bei manchen Computern kann man damit den Programmablauf stoppen. Vorsicht: Bei anderen Computern wird damit das gesamte Programm im Speicher gelöscht! In diesem Fall sollte man ESCAPE oder ein anderes Wort benutzen.

★ CLS (engl. **cl**ear the **s**creen = Bildschirm räumen): Läßt alles verschwinden, was sich auf dem Bildschirm befindet.

★ DATA (Daten): Liste von Dingen, wie Wörter oder Zahlen, die vom Computer in Variablen gespeichert werden. Siehe dazu auch READ.

DIM: Sagt dem Computer, wieviel Speicherraum er für eine Variable zur Verfügung stellen soll. DIM A$(5,4) bedeutet z. B., daß für die Variable fünf Reihen zu je vier Spalten benötigt werden.

★ EDIT: Ermöglicht die Abänderung einer Programmzeile, ohne daß die gesamte Zeile nochmals eingegeben werden muß.

★ END (Ende): Teilt dem Computer das Programmende mit. Bei manchen Computern ist der END-Befehl stets erforderlich, andere Computer, wie z. B. der Sinclair, benötigen diesen Befehl nicht.

FOR...NEXT: Bewirkt einen Rücksprung im Programm und die Wiederholung aller Anweisungen innerhalb einer Schleife für eine bestimmte Anzahl von Durchläufen.

GOSUB: Bewirkt, daß der Computer das Hauptprogramm verläßt und ein Unterprogramm zur Bewältigung einer speziellen Aufgabe abarbeitet.

GOTO (gehe zu...): Bewirkt, daß der Computer zu einer anderen Programmzeile springt.

IF...THEN (wenn...dann): Vergleicht einzelne Daten (z. B. Zahlen, Wörter oder Variablen-Inhalte) und veranlaßt bestimmte Dinge, die von den Ergebnissen des Vergleichs abhängen. Bei manchen Computern kann das THEN entfallen.

INPUT (Eingabe): Damit erfragt der Computer Daten von außen, während das Programm abläuft.

INT (englische Abkürzung für integer = ganze Zahl): Verwandelt eine Dezimalzahl (Kommazahl) in eine ganze Zahl, indem alle Ziffern rechts vom Komma (Punkt) weggelassen werden; z. B. INT(3.40)=3. (Der Computer kennt – genau wie der Taschenrechner – nur die englische Schreibweise von Dezimalzahlen, d. h. mit Punkt statt mit Komma.)

★ LEFT$ (engl. left = links): Bewirkt, daß der Computer sich mit dem linken Teil eines Wortes befaßt. LEFT$ (A$,4) bedeutet z. B.: Nimm von links die ersten 4 Buchstaben von A$.

LEN (englische Abkürzung für length = Länge): Gibt die Länge eines Wortes an, d. h. die Anzahl der Buchstaben in einer Variablen.

LET (es sei): Kennzeichnet einen Platz im Speicher und belegt ihn mit Information, z. B. LET N=4 oder LET B$="KATZEN". Bei den meisten Computern kann dieses Wort entfallen.

★ LIST (auflisten): Läßt das Programm auf dem Bildschirm erscheinen.

★ MID$ (englische Abkürzung für middle = Mitte): Bewirkt, daß der Computer sich mit dem mittleren Teil eines Wortes befaßt. MID$(A$,4,3) bedeutet z. B.: Nimm drei Buchstaben von A$, beginnend mit dem vierten Buchstaben.

NEW (neu): Löscht das Programm aus dem Speicher des Computers, um Platz zu machen für ein neues.

★ NEWLINE-Taste (Zeilenwechsel-Taste): Sagt dem Computer, daß eine Programmzeile oder eine Eingabe vollständig eingegeben wurde. Bei manchen Computern heißt diese Taste RETURN oder ENTER.

★ PLOT (zeichne): Veranlaßt den Computer zum Zeichnen eines Bildpunktes; z. B. bedeutet PLOT (X, Y): Zeichne den Bildpunkt mit den Koordinaten X (nach rechts) und Y (nach oben).

PRINT (drucke): Bewirkt, daß der Computer etwas auf dem Bildschirm sichtbar macht.

★ READ (lies): Veranlaßt den Computer, die Information in einer DATA-Zeile zu lesen und in einer Variablen zu speichern. Vergleiche DATA.

★ READY (fertig, bereit): Manche Computer zeigen dadurch an, daß sie bereit sind, weitere Anweisungen auszuführen.

REM (engliche Abkürzung für remark = Bemerkung): Der Computer ignoriert Zeilen, die mit REM beginnen, er gibt sie jedoch in der Programmliste aus. REM-Zeilen dienen dazu, Programmteile mit einer entsprechenden Bemerkung inhaltlich zu kennzeichnen.

RETURN (kehre zurück): Bewirkt am Ende eines Unterprogramms, daß der Computer zum Hauptprogramm zurückkehrt und die Stelle bearbeitet, die an der Reihe gewesen wäre, bevor er das Hauptprogramm verlassen hat. Vergleiche GOSUB.

★ RIGHT$ (engl. right = rechts): Bewirkt, daß der Computer sich mit dem rechten Teil eines Wortes befaßt. RIGHT$ (A$,4) bedeutet z. B.: Nimm die vier rechten Buchstaben von A$.

★ RND (englische Abkürzung für **r**a**nd**om = zufällig): Der Computer wählt eine Zufallszahl.

RUN (lauf): Bewirkt, daß der Computer das Programm ausführt.

SQR (englische Abkürzung für **sq**uare **r**oot = Quadratwurzel): Bewirkt, daß der Computer die Quadratwurzel aus einer Zahl zieht.

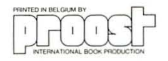
STEP (Schritt): Wird bei FOR...NEXT-Schleifen verwendet und gibt an, mit welchem neuen Wert der Zahlen-Variablen wiederholt werden soll.

STOP (Halt): Bedeutet, daß das Programm vorzeitig abgebrochen werden soll.

★ UNPLOT (Gegenteil von PLOT): Bewirkt das Löschen eines Bildpunktes.

# Register





PRINTED IN BELGIUM BY
proost
INTERNATIONAL BOOK PRODUCTION

# CHIP Umfassende Information über die ganze faszinierende Welt der Mikrocomputer!

## Das CHIP-KOMPLETT-PROGRAMM bietet komplette Information:

### Mit CHIP, dem Mikrocomputer-Fachmagazin

Jeden Monat neu:
Marktübersichten, Kaufberatung, Tests, Anwendungsbeispiele aus der Praxis, Problemlösungen, Entscheidungshilfen, Hintergrundinformationen, Tips und Tricks und mit der CHIP-Börse – dem Kleinanzeigenmarkt.

### Mit CHIP-WISSEN, den Serien-Büchern, wie:

Von der passiven zur aktiven Computerei

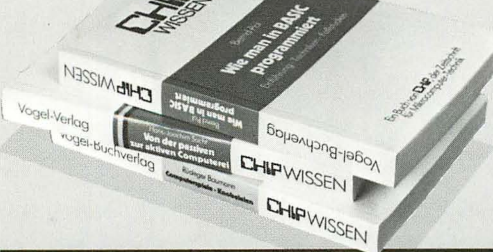
Vom Problem zum Programm

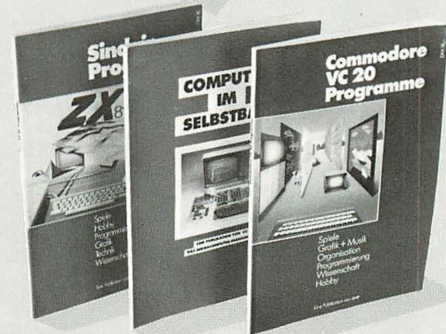
Basic – Versionen im Vergleich

Spannende Simulationen mit dem ZX Spektrum

Was der ZX Spektrum alles kann

Wie man in Basic programmiert




### Mit den periodisch erscheinenden CHIP-SPECIALS, wie:

VC 20-Programme

Computer für Heim und Freizeit

Computer im Selbstbau

ZX 81 – Sinclair-Programme

Computer-Katalog

Computergrafik




**Wer mehr über dieses komplette Info-System erfahren möchte, schreibt an den CHIP Leser-Service, Postfach 6740, 8700 Würzburg.**

# Spaß mit Computern

Computer sind kein Hexenwerk. Sie gehören inzwischen zum Alltag in Betrieben, in Schulen und für viele auch in der Freizeit. Sie müssen nicht Experte sein oder sich erst mühsam spezielle Fachkenntnisse erwerben, wenn Sie diesem elektronischen Hobby nachgehen wollen. Die reich illustrierten Bände dieser Reihe geben Ihnen in leichtverständlicher Sprache einen Einblick in Technik und Funktionsweisen und zeigen Ihnen die vielfältigen Anwendungsmöglichkeiten von Heimcomputern. Damit können Sie Ihr Interesse an moderner Technik aktiv nutzen – und Sie werden sehen, daß das auch noch Spaß macht!

Eine Einführung in Nutzen und Gebrauch von Mikrocomputern: Bauteile, Funktionen, Tips für Benutzung und Kauf, Erläuterung der wichtigsten Fachbegriffe.







Machen Sie Ihren Heimcomputer zum nützlichen Helfer in Alltag und Freizeit! Dieses Buch gibt Ihnen viele Anregungen, Hinweise und Programmbeispiele.

So lernen Sie programmieren – auch ohne Computer! Ein Grundkurs in BASIC: Schritt für Schritt und mit vielen praktischen Beispielen und Programmierübungen.







Mit diesem Buch wird Ihr Heimcomputer zum vielseitigen Spielpartner. Es zeigt Ihnen, wie Sie mit ihm oder gegen ihn spielen – und gewinnen!

ISBN 3-473-**35602**-6 (Otto Maier Verlag)

ISBN 3-8023-**0765**-8 (Vogel-Verlag)